

Erwin Neutzsky-Wulff

# PROGRAMMERING med COMMODORE BASIC



BORGEN



# Oversigt over koder, der er benyttet i bogen for at øge læseligheden

|          |                                   |
|----------|-----------------------------------|
| fA       | Tallet A med CTRL holdt nede      |
| <u>A</u> | Tasten A med SHIFT holdt nede     |
| cA       | Tasten A med COMMODORE holdt nede |
| h        | Home                              |
| <u>h</u> | Clr                               |
| m        | Mellemrum                         |
| n        | Markør op                         |
| s        | Markør ned                        |
| ø        | Markør til højre                  |
| v        | Markør til venstre                |
| p        | Potens                            |
| i        | Insert                            |

Bilag til: Erwin Neutzsky-Wulff: Programmering med COMMODORE BASIC

BORGEN 1983

2. oplag

# Adresser på systemvariabler

- 51-52 Oplagring af skærms startadresse (32)
- 55-56 Oplagring af skærms startadresse (32)
- 197 Berørt tast (40)
- 209-210 Begyndelse på markørs linje (31)
- 211 Markørs plads på linje (31)
- 214 Markørs linjenummer (26)
- 646 Skrivefarve (22)
- 650 Repetér på taster (24)
- 774 Sikring mod listning (23)
- 788 RUN STOP ud af kraft (31)
- 4096-4601 Skærm i udvidet (40)
- 7680-8185 Skærm i uudvidet (23)
- 32768-36863 ROM-karaktergenerator (31)
- 36865 Lodret billedhold (34)
- 36869 Oplagring af karaktergenerators startadresse (32)
- 36874 Bas (27)
- 36875 Alt (27)
- 36876 Sopran (27)
- 36877 Støj (27)
- 36878 Lydstyrke (27)
- 36879 Skærmfarve (22)
- 37137 Output-register A (38)
- 37139 Dataretningsregister for A (38)
- 37150 RESTORE ud af kraft (31)
- 37152 Output-register B (38)
- 37154 Dataretningsregister for B (38)
- 37888-38393 Farvehukommelse i udvidet (40)
- 38400-38905 Farvehukommelse i uudvidet (23)

Tal i parentes hentyder til lektioner.

Når du pøker 36869, skal du trække 48 fra værdien, hvis du har 16K tilsluttet. Husk også at tage højde for flytning af skærm og farvehukommelse!



PROGRAMMERING  
MED  
COMMODORE  
BASIC

*Af samme forfatter:*

Dialog, roman 1971

Udstillingsbilleder, digte 1972

Tidsmaskinen, essay 1972

Seks hundrede seksogtres, digte 1972

Adam Harts opdagelser, roman 1972

Kærlighed, digte 1972

Etik, essay 1973

Parentes begynd, digte 1973

Anno Domini, roman 1975

Gud, roman 1976

Victor Janis og søn, roman 1977

Adam Hart og sjælemaskinen, roman 1977

Oiufael, roman 1977

Den treogtredvte marts, roman 1977

Havet, roman 1978

Indsigtens sted, roman 1980

Menneske, roman 1982

Mikrodatamaten. Programmering og Anvendelse,  
en bog om ZX 81 BASIC, 1982

*Erwin Neutzsky-Wulff*

PROGRAMMERING  
MED  
COMMODORE  
BASIC

BORGEN

Copyright ©1983 Erwin Neutzsky-Wulff

Alle rettigheder, herunder retten til erhvervsmæssig udnyttelse af bogens programeksempler, forbeholdes. Ingen del af denne bog må gengives, mangfoldiggøres ved fotokopiering eller på anden måde, lagres i databank, forvandles eller transmitteres med brug af elektroniske eller mekaniske midler uden skriftlig tilladelse fra forlaget.

Omslag: Claus Deleuran

Illustrationer: Jesper Deleuran

Trykt hos Narayana Press

ISBN 87-418-5514-0

2. oplag

*Til Adam, Elisabeth, Grace, Ionès, Lea, Lilith,  
Marcus, Oiufael, Palle, Rachel og Rebekka.*

# INDHOLD

## FORORD 7

### FØRSTE DEL:

#### VARIABLER OG PROGRAMMER

1. Variabler 13
2. Tastaturet 19
3. Programmer 24
4. Farveprogrammer 29
5. Goto 34
6. Input 39
7. If 44
8. Funktioner 48
9. Relationer 52
10. Logiske operationer 57

### ANDEN DEL:

#### DATA OG SÆT

11. For-next-løkker 65
12. Strengdeling 69
13. Gosub 73
14. Sæt 77
15. Data 81
16. Strengsæt 85
17. Definerbare funktioner 90
18. Andre funktioner 94
19. Save 99
20. Spil 103

## **TREDJE DEL:**

### **BILLEDE OG LYD**

- 21. Bits og bytes 111
- 22. Systemvariabler 117
- 23. Skærmen 122
- 24. Bevægelse 127
- 25. Betingede udtryk 132
- 26. Skærmkoder 137
- 27. Lydeffekter 143
- 28. Toner 148
- 29. Melodier 153
- 30. Musik 157

## **FJERDE DEL:**

### **KARAKTERER OG GRAFIK**

- 31. Karakterer 165
- 32. Definerbare karakterer 170
- 33. Nyt karaktersæt 174
- 34. Grafik i spil 179
- 35. Bit-logik 182
- 36. Højopløselig grafik 186
- 37. Grafisk afbildning 190
- 38. Joystick 194
- 39. Flerfarvede karakterer 199
- 40. Sidste tips 204

## **APPENDIKS**

- Adresser og koder 209
- Skærmkoder 212
- ASCII-koder 214
- Stikordsregister 217

# FORORD

Det var nærmest ved et tilfælde, jeg begyndte at interessere mig for programmering. Jeg skulle skrive en roman, der foregik i miljøet, og så var der netop kommet en »personlig« datamat til 2000 kr. frem, som jeg tog hjem og begyndte at eksperimentere med.

Et lidt usædvanligt motiv, måske, men det betyder jo mindre. Nogles motiv vil uvægerlig være at komme ind på et arbejdsmarked, måske i virkeligheden det eneste, hvor der ikke er nogen overhængende fare for arbejdsløshed. Andre vil se et samfund vokse op, der er uforståeligt for den, der intet kendskab har til disse ting, og frygte, at han ikke længere skal være i stand til at deltage i beslutningsprocesserne, med andre ord, at demokratiet ramler sammen om ørerne på os og erstattes af et middelalderligt ekspert-vælde.

Hvad motiverne end kan være, opdagede jeg hurtigt, at der var et enormt, uopfyldt behov for litteratur, der kunne føre elektroniske analfabeter som undertegnede ind i programmeringens verden. Denne mistanke bestyrkedes, da Borgens forlag bestilte en lærebog i programmering med den lille ZX81, som først havde fanget min interesse, som udgangspunkt, og denne bog, der kom til at hedde MIKRO-DATAMATEN – PROGRAMMERING OG ANVENDELSE, blev en ubetinget salgssucces. Det var nærliggende at følge den op med en bog med udgangspunkt i COMMODORE BASIC, som ikke mindst tales af den uden sammenligning mest solgte farvedatamat VIC 20, både herhjemme og internationalt.

Denne bog handler ikke om elektronik, det forekommer

mig i denne forbindelse mindre relevant, hvordan en chip virker. Derimod anser jeg det for noget nær livsvigtigt for enhver at vide, hvordan man taler med disse bæster. Ligesom jeg vil påstå, at det ikke er muligt at lære at læse uden en bog, tror jeg heller ikke, man kan lære programmering effektivt uden fast adgang til en maskine. Den skal med andre ord stå på ens skrivebord. Det næste skridt er så at prøve at hitte ud af den. Den vil næsten uvægerlig »gå på« BASIC, og så får man altså lært programmerings-verdenssprog. Men hvordan? Ikke af den medfølgende brugsanvisning, der selvfølgelig hverken kan eller skal være et kursus i BASIC. Heller ikke af generelt-teoretiske afhandlinger af den type, man kan låne på biblioteket, der med alle deres paragraffer og ligninger ligner min barndoms matematikbøger.

Måske vil en og anden indviet ærgre sig over min »kan-du-se-den-knap-der«-stil. Så må man forstå, at jeg forklarer tingene sådan, som jeg ønsker, nogen havde forklaret mig dem, dengang jeg skvattede rundt i funktionerne som en baby i Salmonsens.

Desuden er jeg dovent anlagt. Videbegær er ligesom ikke helt nok. Det er ligeledes ud fra egen erfaring, at jeg har forsynet min bog med »appetitvækkere«, hvoraf en er, at du kan starte totalt blank på første side og, før du er færdig, vil være i stand til at frembringe spil, der i kraft af din erhvervede viden om skærmkontrol, lyd og definerbare karakterer, programmering af joystick m.m.m. er fuldt på højde med, hvad du troede, du var nødt til at betale 75 kroner for – stykket!

Nogle af spillene har jeg naturligvis allerede lavet og listet i bogen. Du kan altså med det samme komme til at gætte tal, ord og farvekombinationer, spise prikker og selv blive spist. I andre af bogens spil kan du skaffe dig en vis begrænset erfaring som bl.a. bankmand, biavler, biolog, boghand-



ler, bombeflyver, børsspekulant, foredragspublikum, general, græker, hukommelseskunstner, komponist, kunstmaler, matematiker, musiker, månepilot, racerkører, ridder, rumkaptajn, skarpskytte, skiltemaler, skraldemand, spiller, spion, statistiker, storvildtjæger, taxichauffør, tennisspiller, økolog og ørkenfarer.

VIC 20 *kan* meget. I denne bog laver vi alt fra strengdeling til højopløselig grafik på den Aldeles Uudvidede Maskine. Der er altså mange muligheder, før du skal til at gribe i lommen igen, og det syntes jeg også lige, du skulle vide.

Hvad du end vil med din maskine, tror jeg, du kan finde noget, du kan bruge, i denne bog. Hvordan får du mest ud af farverne? Hvordan laver du beregningsprogrammer? Undervisningsprogrammer, oversættelsesprogrammer? Taler med maskinen, katalogiserer din pladesamling, gemmer programmer, regner i totalssystemet? Poker systemvariabler, får ting til at flytte sig på skærmen, tegner på skærmen? Arbejder med betingede udtryk, lydeffekter, populær og klassisk musik, laver dine egne bogstaver eller små rumskibe eller hvad du vil, laver grafisk afbildning i højopløselig grafik? Regner med bit-logik, oplagrer data på bånd . . .?

Og først og sidst: Alt er forståeligt, og alt svært bygger på noget let, du allerede har forstået. Hvis du ikke tror mig, så blad om på første lektion. Var det til at forstå?

Jamen så fortsæt bare.

Hilsen

Erwin Neutsky-Wulff

PS. En varm tak til Commodore Data i Horsens og København for deres velvillige bistand under bogens udarbejdelse.



*FØRSTE DEL*

# *Variabler og programmer*



# LEKTION 1

## VARIABLER

\*\*\*\* CBM BASIC V2 \*\*\*\*

3583 BYTES FREE

READY.

Sådan står der på skærmen, når vi tænder datamaten. Lad os starte med at se på, hvad det betyder.

BASIC er det sprog, maskinen taler, og som vi derfor må lære for at kunne tale med og bruge den. Det er det, der er denne bogs formål. Der findes også andre sprog for datamater, men BASIC er det uden sammenligning mest udbredte. De maskiner, der kun taler ét sprog, taler næsten altid BASIC. De, der taler flere, taler også BASIC. Det er en slags datamaternes engelsk, verdenssproget for computere.

CBM BASIC V2 er en dialekt af dette sprog. Når et firma fremstiller datamater i større stil, har det gerne sin egen dialekt af BASIC. Det hænger blandt andet sammen med den bestemte måde, dette firma bygger sine maskiner på. Jamen kan denne bog så lære os at tale med andre end firmaet Commodores maskiner? Ja, på samme måde som en sjællænder forstår en fynbo. Skifter man til en anden maskine, skal man lige vænne sig til den nye dialekt, præcis ligesom hvis en sjællænder flyttede til Fyn. Men sæt, han slet ikke kunne dansk?

Med skiltet CBM BASIC V2 angiver maskinen altså for os, hvilket sprog og hvilken dialekt den taler.

Hvad nu med 3583 BYTES FREE?

Hvis du har prøvet en billig lommeregner, ved du, at hvis vi skal regne ud, hvad  $12 \cdot 34 + 56 \cdot 78$  er, må vi først lade den regne ud, hvad  $12 \cdot 34$  er, og skrive resultatet ned, og så senere lade den lægge det til resultatet af den anden udregning. Den kan ikke huske det første resultat, mens den regner det andet ud. Det kan til gengæld lommeregnere med hukommelse, og det kan en gigant-lommeregner som VIC 20 naturligvis også. Men hvor meget kan den så have »i hovedet« på en gang, hvad er dens hukommelseskapacitet? Den måler vi i BYTES, og hvad dette helt præcis betyder, skal vi vende tilbage til i en senere lektion. Her skal vi blot lidt løst konstatere, at en BYTE cirka svarer til et bogstav. Hvis du altså skal have maskinen til at huske en tekst på 100 bogstaver, skal du bruge godt og vel 100 bytes. 3583 BYTES FREE er maskinens måde at fortælle os, hvor meget hukommelse vi har til rådighed, som er til »fri« afbenyttelse for os, nemlig 3583 bytes. Denne hukommelse bliver så efterhånden »optaget«, når vi sætter datamaten til at huske forskellige ting.

Endelig står der READY, »parat«. Det vil sige, at maskinen er parat til at snakke med os og modtage ordrer. Den er ikke midt i en eller anden gigantisk udregning, som den helst ikke vil forstyrres i, den har faktisk ikke noget at lave. Lad os sætte den i arbejde.

VIC kan altså huske. Udmærket, vi sætter den til at huske nogle tal for os. Den kan få lov til at huske vores telefonnummer: 123456. Hvordan gør vi nu det? Ja, vi kunne simpelt hen indtaste det, men hvad så, når vi skulle have fat i det igen, og det helst skulle være vores nummer og ikke et af de andre hundreder, vi havde tastet ind i maskinen? Ja, så skulle der gerne være navn på. Så vi skriver i stedet

PETER=123456

Lagde du mærke til den lille klods, der står og blinker? Det er *markøren*. Den kan altid vise os, hvor vi er på skærmen, det vil sige, det næste bogstav eller ciffer (den næste *karakter*), vi skriver, vil komme til syne der, hvor markøren nu er.

Hvad nu? Der synes ikke rigtig at ske noget. Nej, for maskinen kan jo ikke vide, om du er færdig. Måske kommer der flere cifre. Du må altså forklare maskinen, at der ikke kommer mere, at din ordre er færdig, og at den gerne må udføre den. Det gør du altid med tasten RETURN. Læg mærke til, at du ikke skal *skrive* RETURN. Det er en fast tast, der bare skal trykkes ned.

Nu kom skiltet READY igen frem på skærmen. Ordren er udført, maskinen husker dit telefonnummer, og den er parat til at modtage din næste ordre. Læg mærke til, at der stadig står 3583 BYTES FREE. Dette er selvfølgelig allerede en gammel oplysning, men den forsvinder ikke fra skærmen, så længe der er plads til, at den bliver stående. Det kan være meget praktisk at have tidligere resultater stående på den måde.

Hvordan får vi nu vores telefonnummer frem igen?

Jo, vi skriver

PRINT PETER

Det betyder skriv (egentlig tryk) på skærmen det, der står opført i maskinen som PETER, eller om du vil, luk skuffen med PETER på op og skriv det tal, du finder i den. Huskede du RETURN? Så skulle tallet gerne vise sig. Og maskinen er igen READY.

Vi kunne godt have nøjedes med

PRINTPETER



Prøv det (husk RETURN). Et mellemrum optager også plads. I denne bog vil vi prøve at lære at undvære dem, også

fordi det er almindelig praksis, og du ellers vil få svært ved at forstå de ordrer, andre har skrevet.

Vi kunne også have nøjedes med

$P=123456$

I al fald indtil vi skal gemme nummeret på en person, hvis navn også begynder med P. Pouls nummer kan så blive oplagret som

$PO=789012$

eller

$P2=789012$

Når vi ikke har brug for at huske Peters telefonnummer mere, men i stedet skal huske, hvad vi har sat parkeringsuret på, kan vi skrive

$P=1230$

Det vil sige, egentlig skriver vi  $P=123\emptyset$ . Vi bruger  $\emptyset$  som nul for at skelne det fra O som i Oda. Det var altså klokken halv et. Nu er skuffen P selvfølgelig blevet tømt, og der er lagt et andet tal ned i den. P kan stå for eller indeholde hvad som helst. Det er en *variabel*. Det tal, den *værdi*, vi anbringer i den, siger vi, vi *tilskriver variablen*. I ovenstående eksempel har vi altså tilskrevet variablen P værdien 1230.

Hvad nu, hvis vi skal huske telefonnummeret på to personer ved navn Peter? Prøv

$PETER1=123456$

Og (efter RETURN, selvfølgelig)



PETER2=789012

Skriv nu (igen efter RETURN)

PRINTPETER1

Av. Vi fik nummeret på PETER2 i stedet for. Hvorfor? Fordi maskinen kun læser de to første karakterer i variabelnavnet og følgelig ikke kan skelne mellem PETER1 og PETER2, hvorfor den opfattede PETER2 som en ny værdi af PETER 1. Derimod havde P1 og P2 selvfølgelig været i orden. Dette er altså endnu en fordel ved at bruge forkortede variabelnavne, vi bliver lettere opmærksomme på, hvis vi bruger den samme variabel to gange. I virkeligheden ville man naturligvis aldrig oplagre telefonnumre i en datamat på denne upraktiske måde. Vi skal senere se, hvordan man gør.

Læg mærke til, at vi nu har mistet den øverste del af den oprindelige tekst. Der var ikke plads til den længere.

Måske er du kommet ubesværet gennem disse første eksempler. Men du har muligvis også haft problemer undervejs. Prøv

PRUNTPETER

Maskinen svarer med

?SYNTAX

ERROR

READY.

*Syntaksen* er BASIC-sprogets regler. Når du ikke følger dem, fortæller maskinen dig det:

*Syntaks-fejl!* Og så er den for resten parat til, at du indtaster det rigtige . . .

Du kunne selvfølgelig også med det samme have rettet din fejl. Skriv

PRU

Nu kan du få U væk ved at bruge slettetasten INST DEL. Tryk på den, og U forsvinder. Skriv videre, til du har det rigtige PRINTPETER, og tast RETURN, og du får dit 789012. DEL betyder DELETE, slet. INST står for INSERT, en anden funktion på samme tast, som vi vender tilbage til.

Lad os til sidst indtaste nogle væddeløbsheste og deres odds. Vi har

TAROK=3  
TROFAST=97

og

SPRINTER=5

Hvad nu? Syntaksfejl igen? Javist, der er et S, et E og et R foruden et = for meget i PRINT. Maskinen opfattede din variabel som en forkert stavet ordre. Se, det var jo ikke sket, hvis du havde brugt TA, TR og SP, vel? Flere variabler i næste lektion!

NB! Prøv

PRINT123456

## LEKTION 2

### TASTATURET

Skriv

PRINTA

Da variablen A ikke er tilskrevet nogen værdi, skriver maskinen 0. Men hvorfor skrev den i grunden ikke bogstavet »A«?

For at skelne variabelnavne fra *streng*, det vil sige udtryk, der indeholder bogstaver, som »VIC 20« og »DATAMAT« eller »DER ER ET YNDIGT LAND«, forlanger syntaksen, at *streng* skrives i gåseøjne. Skal vi have skrevet et »A« på skærmen, skal vi altså skrive

PRINT"A"

Gåseøjnene får du ligesom på en almindelig skrivemaskine med skiftenøgle. Hold SHIFT (lige til venstre for Z) nede og tast 2.

*Streng* kan naturligvis også gemmes, nemlig i *strengvariabler*, ligesom tal eller »numeriske« størrelser gemmes i *numeriske variabler*. Navne på strengvariabler kræver endvidere et dollartegn efter sig. Eksempelvis er altså

123 et tal

»VIC 20« en streng

A en numerisk variabel

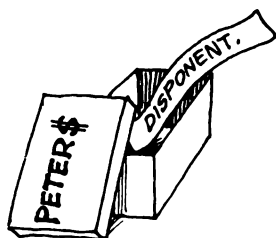
A\$ en strengvariabel

Skriv

PETER=41

og

PETER\$="DISPONENT"



Peters alder kunne opbevares i en numerisk variabel, fordi den var et tal, mens hans stilling, som var bogstaver, altså en streng, krævede en strengvariabel. Skriv nu

PRINTPETER\$PETER

Er du med?

Endelig kan et variabelnavn have procenttegn efter sig. Så er det en *heltalsvariabel*. Det er klart, at tallet 12.3456789 kræver mere plads end tallet 12. I variabelen (skuffen) A er der afsat plads til alle de mange decimaler. Hvis det så kun er 12, vi skal oplagre, spilder vi al den gode plads. Skriver vi nu i stedet for

A=12

ordren

A%=12

har vi dermed valgt en mindre rummelig og mere pladsbesparende skuffe. Den kræver simpelt hen mindre plads i hukommelsen. Til gengæld har vi så med denne formulering afskåret os fra at lade variabelen stå for et tal, der ikke er helt. Prøv

A%=12.3456789

og

```
PRINTA%
```

Variablen har »glemt« decimalerne. Prøv så

```
PRINT A
```

Ja, A er stadig 0. A% er en selvstændig variabel, der har lige så lidt med A at gøre som med A\$. Hvad er for resten A\$? Skriv

```
PRINT A$
```

Maskinen skriver ingenting, for variabelen er ikke tilskrevet nogen værdi. Strengen er »tom«. Skriv nu

```
CLR
```

og undersøg igen værdierne af PETER, PETER\$ og A%.

Strengvariablen er tømt, de numeriske nulstillede. Det er nemlig denne ordres funktion. CLR står for CLEAR, »rens« eller tøm alle variabler!

Ikke alene kan maskinen skrive bogstaver, den kan også skrive i flere farver. Du kan finde dem under tasterne 1-8:

- 1: BLK=BLACK=SORT
- 2: WHT=WHITE=HVID
- 3: RED=RØD
- 4: CYN=CYAN=LYSEBLÅ
- 5: PUR=PURPLE=VIOLET
- 6: GRN=GREEN=GRØN
- 7: BLU=BLUE=BLÅ
- 8: YEL=YELLOW=GUL

Hvis du prøver at taste dem, får du bare tallene. Du skal bruge CTRL (CONTROL = kontrol) og holde denne tast nede, mens du taster farven. Prøv at holde CTRL nede, mens du taster 3. Nu vil alt, hvad du skriver på skærmen, blive rødt, til du igen skifter farve. Skriv på denne måde VIC 20, dit navn, osv. Du kan nu fremhæve enkelte ord ved at skrive dem i en anden farve, men du kan også »vende dem om«, så de skrives f.eks. hvidt på rødt i stedet for rødt på hvidt. Dertil bruges tast RVS (REVERSE = omvend) ON (»slået til«) og RVS OFF (fra). Igen skal du bruge kontroltasten. Hold altså CTRL nede, mens du taster RVS ON.

Nu vil alt, hvad du skriver, blive »omvendt«. Mellemrummet, der normalt er »ingenting«, vil skrive en klods i den farve, du har valgt. Så længe du holder den lange tast nede, vil den trække en bjælke på skærmen i denne farve. Lige nu trækker den måske en rød bjælke. Slip nu tasten og hold CTRL nede, mens du taster 6 (GRN). Var det svært? Prøv at skrive VIC 20. Det blev hvidt på grønt, ikke?

Men VIC kan mere end at skrive bogstaver. Prøv at holde SHIFT nede og taste S. Du får et hvidt hjerte på grøn baggrund. Lidt unaturligt, måske, men det kan vi nemt rette. Hold CTRL nede og tast RVS OFF. Nu fik vi »normal« skrift. Skift nu farve til rød (3 med CTRL nede). Ikke sandt?

Det, du fik, kan du finde under S-tasten på højre side. Tasten til venstre for SHIFT, der bærer Commodores bomærke (C=), holder du nede, når du skal have dem til venstre. Hold COMMODORE nede og tast B. To små røde kvadrater toner frem. »Commodore«-grafikken er måske knap så morsom som SHIFT-grafikken. Førstnævnte egner sig bedst til statistik, osv. Sidstnævnte kan vi bruge til at lege med. Prøv at trykke klør, spar, ruder og hjerter i de rigtige farver.

Hvor får vi nu de små bogstaver? Hertil kræves, at maskinen bringes i en tilstand, hvor den »læser« tastaturet på en anden måde. Hold SHIFT nede, mens du trykker på C=-tasten. Læg mærke til, hvordan hele skærbilledet ændre-

de sig. I denne tilstand får du »normalt« små bogstaver og store med SHIFT, ligesom på en almindelig skrivemaskine.

Med C= holdt nede får du stadig »venstre«-grafikken. Højre-grafikken kan ikke nås i denne tilstand. Det er den mere »forretningsmæssige« side af tastaturet, vi her har fået frem. Vi skynder os at skifte tilbage med endnu et tryk på C= med SHIFT i bund.

Måske er det nu nærmest skærmen, der trænger til at blive »renset«. Hold SHIFT nede og tryk på CLR HOME-tasten. CLR står for CLEAR SCREEN, slet skærmen. Hvis du blot trykker på tasten uden SHIFT, er det HOME-funktionen, du får fat i. Markøren vender »hjem«, det vil sige til øverste venstre hjørne af skærmen. Det gør den selvfølgelig også med CLR, men den fjerner altså samtidig skærmbilledet. Pas på ikke at forveksle CLR-tasten, der sletter *skærmen*, med *ordren* CLR, der sletter *variablerne*!

Du kan også flytte markøren andre steder hen ved hjælp af de to markørtaster mærket CRSR (CURSOR-markør). Pilene viser, hvad vej de flytter markøren, men skal du *op* og *til venstre*, skal du samtidig holde SHIFT nede. Trykker du en enkelt gang, flytter markøren et enkelt felt. Holder du tasten nede, kan du få den til at fare af sted, til du slipper.

På denne måde kan du rette overalt på skærmen. Skriv skærmen fuld. Bestem dig så for at rette et bogstav og lad markøren dække det. Nu behøver du kun at taste det bogstav, der skal erstatte det! Ret lidt rundt på skærmen på denne måde.

Eksperimenter med alt, hvad du har lært om tastaturet! I næste lektion skal vi nemlig i gang med at skrive et rigtigt program . . .

NB! Prøv

```
PRINT2"+2"=5
```

## LEKTION 3

### PROGRAMMER

Skriv ordrerne

```
PETER=41  
PETER$="DISPONENT"  
PRINTPETER$PETER
```

Maskinen skriver som før DISPONENT 41. Men sæt, vi nu ville have den til at gøre det samme igen? Så skulle vi skrive alle ordrerne en gang til. I stedet kan vi sætte ordrerne ind i et system af ordrer, en oversigt over, hvad maskinen skal gøre, bestående af ti eller hundrede ordrer. En sådan ordreliste kaldes et *program*. Vi kunne altså have

```
1 PETER=41  
2 PETER$="DISPONENT"  
3 PRINTPETER$PETER
```

Tallene foran hver ordre er *linjenumre*, der fortæller maskinen, i hvilken orden den skal udføre ordrerne, der nu er blevet *programlinjer* i programmet. Vi behøver altså ikke at taste linjerne ind i den rigtige orden, blot vi forsyner dem med de rigtige linjenumre. Og for at sikre os, at vi kan passe noget ind, vi har glemt, er det smart at lave et mellemrum på f.eks. 10 mellem de enkelte linjenumre. Vi får så

```
10 PETER=41  
20 PETER$="DISPONENT"
```



## 30 PRINTPETER\$PETER

Prøv at skrive programmet op. Vi skal have RETURN efter hver linje. Mellemrummet mellem linjenummer og linjens indhold er ikke nødvendigt, men kan gøre programmet lettere læseligt.

Er du færdig? Der sker ikke noget. Nej, for et program må jo ikke »starte«, før det er skrevet færdigt. Det skal have en særlig ordre for at »køre«, og den hedder

## RUN

Tast altså RUN fulgt af RETURN, og programmet går i gang. Tast RUN igen, og programmet kører endnu en gang. Du har givet maskinen en »fast« opskrift og dermed skrevet dit første program. Prøv nu

## RUN 30

Nu skriver maskinen bare 0, for din ordre lød faktisk på, at maskinen skulle køre programmet *fra og med* linje 30, og så fik den ganske enkelt ikke værditilskrivningerne med. Men hvorfor var de der ikke fra sidste programkørsel? Fordi RUN *automatisk sletter alle variabler*. RUN betyder faktisk CLR & RUN! Hvad vil RUN 20 give? Prøv at regne det ud, og se så efter, om du havde ret. Kør programmet endnu et par gange. Nu kan du ikke længere se din PROGRAMLISTE. Skriv

## LIST

og den kommer frem igen. Hvad vil LIST 30 give? Nemlig, linje 30 på skærmen. Prøv LIST 20-30. LIST 20-. LIST -20. Forvirret? Jo:

LIST giver hele programlisten

LIST N (hvor N er et eller andet linjenummer) giver linje N

LIST N- giver programlisten fra og med linje N

LIST -N giver programlisten til og med linje N

LIST M-N giver programlisten fra og med linje M til og med linje N

Det er selvfølgelig noget, du får mere brug for, når dine programmer bliver lidt længere! Skriv igen

## LIST

Nu får du brug for, hvad du lærte om markørtasterne. Ret på denne måde 41 til 51. Men pas nu på! Dette retter kun tallet på *skærmen*. Du har ikke virkelig rettet i *programmet*, hvis du ikke følger rettelsen op med RETURN. Prøv selv!

Gå nu op og ret linjenummer 10 til 15. Tag så en LIST, men pas på, du ikke kommer til at skrive oven i noget andet, så du får LISTY eller noget andet for maskinen uforståeligt. Hov! nu er der både en linje 10 og en linje 15 med præcis det samme indhold. Javist, maskinen har jo en gang fået besked om linje 10, og det glemmer den ikke, medmindre den får besked om *det*. Du kan klare det med at skrive en »tom« linje, dvs. du taster simpelt hen linjenummeret på den linje, du vil af med, fulgt af RETURN. Slip af med 15 på denne måde og bed igen om listen. Nemt nok, ikke?

Du kan selvfølgelig også rette en linje ved at skrive den helt om. En ny linje erstatter altid en tidligere med samme linjenummer.

Men hvad nu, hvis du simpelt hen vil af med noget i en linje? Måske skal maskinen kun skrive værdien af PETER, og ikke af PETER\$? Ja, så skal vi af med PETER\$ i linje 30, og det går faktisk let, vi kan nemlig bruge vores slettetast fra før. Men først skal vi have markøren på plads. Sørg for at anbringe den oven på P i det andet PETER (som om vi lige

havde skrevet det, vi vil af med) og tast så DEL. Ikke blot forsvinder PETER\$ karakter for karakter, linjen trækker sig også sammen, så der ikke bliver noget »hul«. Læg mærke til, at også denne tast »fortsætter«, så længe du holder den nede, så pas på, den ikke løber fra dig! Nu hedder linje 30

### 30 PRINTPETER

Husk RETURN!

Vi kan også gå den anden vej og tilføje noget inde midt i en linje. Så er det, vi skal bruge den tidligere omtalte INSERT (indsæt), som vi får med SHIFT. Lad os f.eks. sige, at vi vil have linjen til at hedde

### 30 PRINT"A:"PETER

måske for at gøre opmærksom på, at tallet er en alder. Ja, så nytter det ikke noget, bare at rette, for vi skal faktisk have linjen til at udvide sig og skaffe plads til yderligere 4 karakterer. Okay, vi rykker igen markøren hen, så den står direkte *efter* det, vi vil ændre på, altså oven i P i PETER. Hold nu SHIFT nede og tast INST 4 gange. Nu blev der plads, og du kan skrive

"A:"

fulgt af RETURN.

Nu vil du måske skrive et andet program, men så skal du først af med det, du har. Det har vi også en ordre til, nemlig

### NEW

Prøv at sætte linjenumre på nogle af de tidligere eksempler og køre dem som programmer. Du kan også bruge f.eks.

CLR i et program. NEW i sidste linje i programmet vil slette det, så snart det er kørt. Eller prøv

```
10 PRINT"VIC 20"  
20 RUN
```

Hvad gør maskinen? Jo, først skriver den VIC 20. Så går den ned til linje 20, der giver den besked på at køre programmet forfra, det vil sige, den skriver igen »VIC 20«, osv. Nu bliver problemet nærmest at standse programmet igen, men til det brug har vi tasten RUN STOP. Ved brug af den får vi beskeden

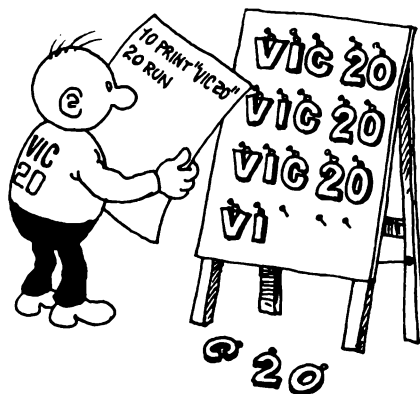
```
BREAK IN 10
```

dvs. »standset, mens jeg var i færd med at udføre linje 10«. På samme måde vil en syntaksfejl i et program give en besked, der gør opmærksom på, i hvilken linje fejlen findes.

Nu, da du ved, hvad et program er, kan vi gå over til nogle lidt mere farverige sådanne i lektion 4.

NB! Prøv

```
10 PRINT"OOOOOOOOOOOOOOOOOOOOOOOO"  
20 PRINT"*****"  
30 RUN
```



## LEKTION 4

### FARVEPROGRAMMER

Skriv

```
10 PRINT"VIC 20"  
20 PRINT"FRA COMMODORE"  
30 PRINT"DEN PERSONLIGE DATAMAT"
```

og kør. Hold CTRL nede og skriv 3. LIST programmet og kør det igen. Javel, men var det nu ikke mere praktisk, hvis f.eks. »VIC 20« og FRA kunne stå med grønt, »COMMODORE« med rødt og *resten med sort*? Lad os prøve. Det kræver i al fald, at farvekontrollerne kommer inden for gåseøjnene, og for nu ikke at blande for meget sammen af det, vi lige har lært, kan vi blot skrive NEW og gå i gang igen. Skriv

```
10 PRINT"
```

Det er nu, vi skal have fat i farvekontrollen, som altså skal være den for grøn. Vi holder CTRL nede og taster 6. Nu fik vi en opadvendt hvid pil på farvet baggrund. Tja, sådan ser farvekontrollerne altså ud inden for gåseøjne. Skriv linjen færdig og fremstil de andre linjer på tilsvarende måde. Linje 20 skal du altså skrive som

```
20 PRINT"FRA
```

mellemrum, CTRL nede, 3, og så resten af linjen. Farvekontrollen behøver altså ikke at ligge i begyndelsen af en linje.

Du fik et pundtegn for rød og et (farvet) kvadrat for sort, ikke? Farvekontrollen vil vi for fremtiden i de programlister, der er aftrykt i denne bog, angive med et lille f foran tallet på den pågældende farvetast. Vores program ville altså komme til at hedde

```
10 PRINT"f6VIC 20"  
20 PRINT"FRA f3COMMODORE"  
30 PRINT"f1DEN PERSONLIGE DATAMAT"
```

Du kan egentlig godt springe mellemrummene mellem linjenummer og linje over. Så snart du beder om programlisten, sætter maskinen dem selv.

Hvad med omvendt skrift? Efter præcis samme opskrift. Prøv at erstatte 20 med

```
20 PRINT"FRA f9COMMODORE"
```

eller

```
20 PRINT"FRA f3f9COMMODORE"
```

Vi kan også fremhæve »PERSONLIGE« i 30:

```
30 PRINT "f1DEN f9PERSONLIGE DATAMAT"
```

Men hov, nu blev »DATAMAT« jo også omvendt. Ja, for vi glemte at »slukke« vores omvender. Linjen skal altså hedde

```
30 PRINT"f1DEN f9PERSONLIGEf0 DATAMAT"
```

Altså:

En farvekontrol gælder i et program, til maskinen støder på en anden.

»Omvenderen« gælder linjen ud.

Mens vi er ved bogens programlister, skal grafikken jo også angives. Her vil et hjerte blive angivet som S, dvs. S SHIFT (skiftet S). Venstregrafikken, som vi får med COMMODORE-tasten, angives med et lille c. cB er altså det grafiske tegn til venstre på B, eller om man vil: B med cCOMMODORE holdt nede. Denne »kode« skulle afværge de fleste misforståelser. Prøv

```
10 PRINT"f3Sf7Qf1W"
20 RUN
```

Læg mærke til, at maskinen automatisk skifter linje efter hver PRINT-ordre. Tilføj nu et , til linje 10. Nu sprang den kun en halv linje. Forsøg endelig med ; Let, ikke? Vil du have det til at gå langsommere, kan du holde CTRL nede. Selvfølgelig kan det gøres på en smartere måde. Det vender vi tilbage til. NEW og

```
10 PRINT"
```

Tryk nu på HOME (altså uskiftet), og et omvendt S kommer frem. Skriv

```
VIC 20"
```

og

```
20 RUN
```

Kør programmet. Maskinen skrev VIC 20 i øverste venstre hjørne, og dermed slut. Og dog. Programmet kører stadig og skal standses med RUN STOP. Det, der sker, er, at HOME, der fremstår i programmet som et omvendt S, bli-

ver ved med at sende maskinen »hjem«, så den bliver ved med at skrive oven i det samme. Udskift nu 10 med

```
10 PRINT"
```

Tryk på CLR (altså skiftet), og et omvendt hjerte kommer frem. Skriv

```
VIC 20"
```

og kør. Nu sletter maskinen også skærmen for hver gennemkørsel, og VIC 20 blinker (meget hurtigt). Vi vil udtrykke HOME med lille h, og CLR med lille h med streg under. De to sidste eksempler ville vi skrive henholdsvis

```
10 PRINT"hVIC 20"
```

```
20 RUN
```

og

```
10 PRINT"hVIC 20"
```

```
20 RUN
```

Vi kan også bruge markørtasterne i et program. Skriv

```
10 PRINT"COMMODORE
```

Tast markør ned og markør til højre. Maskinen skriver omvendt Q og højreklamme. Skriv

```
COMMODORE"
```

I bogen vil vi skrive det som

```
10 PRINT"COMMODOREsøCOMMODORE"
```



s og ø står for »syd« og »øst«. Op og til venstre kommer så til at hedde n og v.

Du skulle nu være forholdsvis bekendt med tastaturet. I lektion 5 skulle vi så kunne tage lidt mere alvorligt fat på programmeringen af VIC 20.

**NB! Prøv**

```
10 PRINT"f3QQQQQQQQQQQQQQQQQQQQQQQQQQ"
20 PRINT"f3f9QQQQQQQQQQQQQQQQQQQQQQQQQQ"
30 RUN
```

## LEKTION 5

### GOTO

Prøv

```
10 PRINTA  
20 A=A+1  
30 GOTO10
```

Hvad gør programmet? Ja, først skriver det værdien af A, som er 0, eftersom vi ikke har tilskrevet den nogen anden værdi. Så lægger den 1 til A. Dette er en lidt speciel form for værditilskrivning. Vi kunne kalde den »værdiforandring«. Linjen betyder: Tag tallet op af skuffe A, læg 1 til og læg det ned igen!

Linje 30 er også ny. GOTO omgår programmets naturlige nummerorden at tage programlinjerne i. I stedet for at gå videre til næste linje (og stoppe, eftersom der ikke er flere) giver linje 30 maskinen besked på at fortsætte kørslen i linje 10. Den skriver altså på ny As værdi (som nu er 1), lægger igen 1 til i linje 20, og så fremdeles. Programmet tæller. Hvorfor kunne vi nu ikke have brugt en linje 30, der hed

```
30 RUN
```

Også her ville vi være kommet tilbage til 10, som jo er programmets første linje. Godt nok, men vi ville være kommet dertil med slettede variabler. A ville igen være 0, og programmet ville skrive lutter nuller. Prøv at erstatte 10 med

```
10 PRINT"h"A
```

Vi kan også få maskinen til at skrive en tabel:

```
10 PRINTA
20 A=A+17
30 GOTO10
```

giver 17-tabellen. Maskinen lirer sin lektie af i en forfærdelig fart. Vi kan naturligvis altid stoppe den undervejs med RUN STOP eller blot sætte farten ned med CTRL. Eller vi kan lave et lidt smukkere program med

```
10 A=A+17
20 B=B+1
30 PRINT"f6"B"f7GANGE 17 ERf3"A
40 GOTO10
```

Eller lad den regne rentesregning. Vi indsætter 100 kroner til 9% rente og får

```
10 A=1983
20 B=100
30 PRINTA":B
40 A=A+1
50 B=B*1.09
60 GOTO30
```

Lad den bare køre lidt. I løbet af 30 sekunder vil den have udskrevet årligt indestående de næste 500 år!

Prøv at standse den ved år 2033. På 50 år er vores 100 kroner blevet til 7435.75. Det kan vist næppe betale sig. Men lad den så køre videre til 2083! Videre? Jeg mener vel forfra igen? Vist ikke. Tast

## CONT

hvilket betyder »continue« = fortsæt, og programmet fortsætter. Nå, nu er den halve million da hjemme. Måske skulle vi lige snuppe 100 år til på den anden side, så vi kan blive hele millionærer. I 2183 har vi nemlig – hvad pokker betyder 3.05702913E+09? Ingen panik! det betyder bare, at kommaet (som altså her er et punktum) skal flyttes ni pladser til højre. Vores indestående er altså kr. 3057029130 eller lidt over 3 milliarder. Mange bække små . . . Som det vil fremgå, kan maskinen regne, og vi kan da også bruge den til ganske almindelig købmandsregning. Prøv f.eks. ordren

PRINT123\*456

Det var lidt lidt at skrive P·R·I·N·T for? Ja, måske, derfor kan du også nøjes med et spørgsmålstegn.

?123\*456

vil virke præcis lige så godt. Regn selv med +, – (minus), \* og / (divideret med). Husk blot, at maskinen *prioriterer*. Dermed menes, at maskinen fornuftigvis er ligeglad med, om du skriver

?123\*456+789

eller

?789+123\*456

-

Den kan aldrig finde på at lægge 789 sammen med 123 og gange resultatet med 456. *Den ganger og dividerer altid, før den lægger sammen og trækker fra.* Men hvad nu, hvis vi virkelig vil

have den til at lægge 789 og 123 sammen, og så gange med 456? Så klarer vi den med parenteser (ikke klammer):

$(789+123)*456$

Pilen på tasten til højre for \* er til »potensopløftning«. I stedet for  $7*7*7*7$  kan vi sige 7 opløftet til fjerde potens, det vil sige 7 ganget med sig selv fire gange. Vi taster  $7^4$ , og i bogen skriver vi

$7^4$

Maskinen opløfter altid til potens, før den indlader sig på nogen af de fire regningsarter. Prøv til sidst

```
10 A=1
20 PRINTA
30 STOP
40 A=A*2
50 GOTO20
```

Maskinen skriver 1 og standser så med beskeden

```
BREAK IN 30
```

Da den kom til STOP, kunne den ikke komme længere. Tast nu CONT, og programmet fortsætter med 2, hvor det igen stopper, og så fremdeles. Somme tider kan man undersøge et program for fejl ved at anbringe STOP-linjer i det. På den måde kan det køres »lidt ad gangen«. Erstat 30 med

```
30 END
```

Denne linje betyder, at programmet er slut (og er derfor overflødig i programmer, der kun skal ende i sidste linje) og

der kommer ingen meddelelse. Vi kan dog stadig få det til at fortsætte med CONT. Slet 30 helt og lad det køre. Ved 4.25352959E+37 standser det med beskeden

?OVERFLOW  
ERROR IN 40

Længere kan maskinen nemlig ikke »tælle«. Så hvis du har mere end 42 billioner billioner billioner i banken, skal du købe en større datamat . . . !

NB! Prøv

```
10 A=1983
20 B=B+100
30 PRINTA": "B
40 A=A+1
50 B=B*1.09
60 GOTO20
```

## LEKTION 6

### INPUT

Prøv igen

```
10 A=A+17
20 B=B+1
30 PRINT"f6"B"f7GANGE 17 ERf3"A
40 GOTO10
```

Hvad nu, hvis vi vil have 18-tabellen i stedet for? Vi kan selvfølgelig rette i programmet, men det ville dog være ulige smartere, hvis det samme program kunne bruges til en hel række tabeller. Prøv

```
5 INPUTC
10 A=A+C
20 B=B+1
30 PRINT"f6"B"f7GANGE"C"ERf3"A
40 GOTO10
```

Det, der bliver lagt til i 10, er nu værdien af variabelen C, og den kommer åbenbart fra linje 5. Men hvordan? Jo, INPUT-linjen betyder simpelt hen, at maskinen skal vente på, at et tal indtastes, og derefter tilskrive det som værdi til variabelen C. Kør programmet. Der kommer et spørgsmålstegn frem på skærmen. Dette er maskinens måde at bede om en indtastning på. Indtast et tal, og programmet kører. Det blotte og bare spørgsmålstegn vil naturligvis forvirre den, der ikke

kender programmet. Derfor ville det måske også være en god idé at indføre en forklarende PRINT-linje, som f.eks.

```
4 PRINT"HVILKEN TABEL"
```

Men vi kan også inkludere den i selve INPUT-linjen og få

```
5 INPUT"HVILKEN TABEL";C
```

Prøv at køre det. Smukt, ikke? Her skulle maskinen altså »fodres« med et tal. Men det kunne også sagtens have været en streng:

```
10 INPUT"DU HEDDER";A$  
20 INPUT"DU FYLDER I 1983";A  
30 PRINT"DU ER FØDT"1983-A;A$
```

Læg mærke til semikolon mellem A og A\$. Uden det ville det være blevet til strengvariablen AA\$. Lad os prøve at slette det. Maskinen siger

```
?TYPE MISMATCH  
ERROR IN 30
```

Dette betyder blot, at vi har prøvet at trække en streng fra et tal, og det kan selvfølgelig ikke lade sig gøre. Ret programmet tilbage og kør igen, men svar nu omvendt på spørgsmålene, altså alder på spørgsmål om navn og omvendt. Maskinen havde ingen vanskeligheder med navnet (måske hedder du virkelig 91), men da du i stedet for det ventede tal indtastede bogstaver, protesterede den

```
?REDO FROM START
```

om igen soldat: Indtast nu bare et tal, og maskinen fortsæt-



ter. Vi kan strengt taget også bruge INPUT som en (noget omstændelig) måde at bremse løbske programmer på. Prøv

```
10 A=1
20 PRINTA
30 INPUTB
40 A=A*2
50 GOTO20
```

Hver gang maskinen når til INPUT-lejren, standser den med et spørgsmålstegn. Tast blot RETURN (der lader B beholde sin værdi, men det er jo lige meget), og programmet tager et brette nøk til!

Hov, hvordan sytten kommer du ud af det her igen? RUN STOP virker ikke. Jo, hvis du holder tasten nede og samtidig trykker på RESTORE! Samtidig bliver markøren blå (hvis den ikke var det i forvejen).

Læg mærke til, at maskinen sagtens kan tage mod flere INPUT i en linje:

```
10 INPUT"DATA";A,B$,C,D
20 PRINT A;B$;C;D
```

Kør programmet med RUN og indtast

```
12 (RETURN)
VIC (RETURN)
34 (RETURN)
56 (RETURN)
```

Du kan spare på RETURNerne ved at indtaste det hele på én gang:

```
12,VIC,34,56 (RETURN)
```

Prøv at indtaste

12,VIC,34 (RETURN)

Spørgsmålstegn angiver, at der er endnu en variabel, der kræver påfyldning. Eller indtast

12,VIC,34,56,78 (RETURN)

Maskinen oplyser:

?EXTRA IGNORED

dvs. »jeg ser bort fra det, der er tilovers«, som der ikke er nogen variabel til!

Vi kan nu få et næsten brugbart renteprogram ud af det:

```
10 INPUT"BELØB";A
20 INPUT"RENTEFOD I %";B
30 C=1983
40 PRINT"f3f9"C"f0f7"A
50 INPUT
60 A=A+A*B/100
70 C=C+1
80 GOTO40
```

Da INPUT-linjen i 50 kun er en pause, kan variabel helt undværes. Vi kan også sætte maskinen til at høre os i tabellerne:

```
10 INPUT"TABEL";A
20 B=B+1
30 PRINT"f3HVAD ER"B"GANGE"A
40 INPUT C
50 PRINT"SVARET ERf9"B*A
60 GOTO20
```

Selvfølgelig er det lidt irriterende at få at vide, hvad »svaret er«, når man lige selv har indtastet det. Bedre ville det være, hvis maskinen skrev BRAVO eller noget lignende, når vi havde svaret rigtigt. Det ville jo imidlertid kræve, at maskinen kunne se, om vi havde svaret rigtigt. Det må vi prøve i næste lektion.

NB! Prøv

```
10 INPUT A
20 B=B+A
30 PRINT B
40 GOTO 10
```

# LEKTION 7

## IF

I lektion 6 havde vi et program, der i princippet så sådan ud:

```
10 INPUT "TABEL";A
20 B=B+1
30 PRINT "HVAD ER "B"GANGE"A
40 INPUT C
50 PRINT "SVARET ER "B*A
60 GOTO 20
```

Det irriterede os, at maskinen ikke havde nogen mulighed for at opdage, om vi svarede rigtigt eller forkert. HVIS vi svarede rigtigt, SÅ skulle maskinen nemlig rose os. HVIS – SÅ hedder på engelsk – og i BASIC – IF – THEN. Vi kunne nu indsætte en linje 45, der hed

```
45 IFC=B*A THEN PRINT "BRAVO"
```



Hvis vi tilføjede

```
46 IFC=B*ATHENGOTO20
```

ville vi endda slippe for 50. Faktisk ville

```
46 IFC=B*ATHEN20
```

være nok, eftersom GOTO er underforstået umiddelbart efter THEN. Faktisk kan vi nøjes med en enkelt linje:

```
45 IFC=B*ATHENPRINT"BRAVO":GOTO20
```

Vi har her fået passet en ekstra linje ind i 45; oven i købet »rider« den på det samme IF – THEN som PRINT-linjen. Alt, hvad der kræves, er *et kolon mellem de to ordrer*, og de er én linje. Maskinen vil udføre alle de ordrer, der står efter THEN *i samme linje*. Eller hvad med

```
10 INPUT"TABEL";A
20 B=B+1
30 PRINT"HVAD ER"B"GANGE"A
40 INPUTC
50 IFC=B*ATHENPRINT"RIGTIGT":GOTO20
60 PRINT"NEJ -"B*A
70 GOTO20
```

Hvad sker der, hvis C ikke er =B\*A? Ja, så går maskinen videre til 60 og skriver NEJ + det rigtige resultat.

*Når betingelsen ikke er opfyldt, går maskinen videre til næste linje.*

Dette program vil selvfølgelig fortsætte til dommedag. Det vil fortsætte med at udfritte os med hensyn til, hvad 97 gange 17 er, og 2345 gange 17 er. Men det behøver det jo egentlig slet ikke. Tilføj

25 IFB=11THENEND

eller bedre

25 IFB=11THEN80

Nu bliver vi nemlig sendt til en endestation, der ligger uden for karrusellen (»løkken«), der så kan hedde

80 PRINTD"RIGTIGE"

Hvad hulen er D? Jo, det er vores »tæller«, der tæller, hvor mange gange vi svarer rigtigt. Derfor skal vi også have udvidet vores 50 til

50 IFC=B\*ATHENPRINT"RIGTIGT":D=D+1:GOTO20

eller på menneske-dansk: Hvis der svares rigtigt, skal maskinen skrive RIGTIGT på skærmen, tælle en på tælleren (»en til rigtig!«) og gå videre til næste spørgsmål (B 1 højere).

Vi har nu fået et program med en meget lang linje og et par meget korte. Kunne vi ikke klistre nogle af de korte sammen til lange med kolon? Jo, sagtens. Og eftersom programmet er færdigt, og der ikke skal passes mere ind, kan vi også spare lidt på linjenumrene. Læg så farver på, og du har noget, der ligner et rigtigt program:

```
1 INPUT"TABEL";A
2 B=B+1:IFB=11THEN5
3 PRINT"f7HVAD ER"B"GANGE"A:INPUTC:IFC=
  B*ATHENPRINT"f3f9RIGTIGT":D=D+1:GOTO2
4 PRINT"f6f9NEJ -"B*A:GOTO2
5 PRINT"f5f9"D"RIGTIGE"
```

Hvorfor kan 60 og 70 ikke klistres bag på 50? Nej, for så bliver de jo også konsekvenser af  $C=B*A$ . Ikke? Desuden er der grænser for, hvor lange linjer må være. Fire skærmlinjer er maksimum.

Endte du med en irriterende lilla skærm? Kombinationen RUN STOP – RESTORE hjælper også her. Du får din »normale« skrivefarve tilbage samt en ren skærm. Men hverken dit program eller dine variabler har lidt overlast!

Vigtigst af alt: Du synes måske, programmet nu er blevet komplet uoverskueligt. Du vil imidlertid ved nærmere studium finde, at det er blevet det stik modsatte. Prøv at læse det:

1. Hvilken tabel?
2. Har vi været tabellen igennem?
3. Stil opgave, modtag svar, undersøg svar; hvis rigtigt, skriv RIGTIGT, tæl 1 rigtig!
4. Hvis forkert, skriv rigtigt svar!
5. Skriv, hvor mange rigtige!

Kan du det, har du taget det første skridt ind i programmeringens mysterier! Det er selvfølgelig ikke overvældende *spændende* programmer, vi har lavet indtil nu. Det vil vi prøve at rette på i næste lektion, hvor vi skal undersøge nogle af de matematiske funktioner, som vi vil finde kan bruges til andet og mere end matematik!

NB! Prøv

```
1 INPUT"TIM,MIN,SEK";T,M,S:PRINT"h"
2 PRINT"h"T"v "M"v "S"v ":S=S+1:IFS=60THENS=0:
  M=M+1
3 IFM=60THENM=0:T=T+1
4 IFT=24THENT=0
5 P=P+1:IFP=121THENP=0:GOTO2
6 GOTO5
```

## LEKTION 8

### FUNKTIONER

$12*34$  er en OPERATION. Den knytter to tal sammen til et: 408. Men det er ikke alle operationer, der virker på to tal. Nogle virker kun på et. De kaldes FUNKTIONER.

Kvadratrod 4 er en sådan funktion. Kvadratroden af et tal er et andet tal, der ganget med sig selv giver det første. Kvadratrod 4 er 2, fordi  $2*2=4$ . Det kan maskinen også regne ud. Prøv

?SQR(4)

og du får 2. Hvad er

?SQR(2)

Læg mærke til, at denne operation er en funktion, fordi den kun virker på ét tal, funktionens ARGUMENT. Dette argument er i *parenteser*. Nu er du måske ikke vild med at uddrage kvadratrødder, men så er der andre funktioner at lege med. Funktioner behøver nemlig ikke at virke på tal, de kan også virke på strenge. Tag

A\$="COMMODORE":?LEFT\$(A\$,3)

Vi får COM på skærmen.

LEFT\$(A\$,3)

betyder nemlig »de tre første karakterer af A\$«. Denne funktion vil vi vende tilbage til i en senere lektion. Der er andre, vi kan drage mere øjeblikkelig nytte af. Prøv



?RND(1)

Du får et tal mellem 0 og 1. Dette tal er *tilfældigt* (random). Kan du se nytten af dette i spil og lignende? Prøv igen. Du får et andet tilfældigt tal. Disse tal er altid mellem 0 og 1. Vi må regne lidt på dem for at gøre dem større og mere brugbare.

I virkeligheden er der selvfølgelig intet tilfældigt ved en datamat. Der er tale om en fast række tal, der er resultatet af så uigennemskuelige udregninger, at de forekommer tilfældige. Vi kan få maskinen til at starte et bestemt sted i følgen med

RND(-A)

hvor A er et eller andet tal. Prøv

?RND(1);RND(1);RND(1)

tre gange efter hinanden, og prøv tilsvarende

?RND(-27);RND(1);RND(1)

Er du med? Og så får vi i øvrigt det underlige E fra lektion 5 igen. Her har vi -08 i stedet for, et negativt tal, altså mindre end 0. Det betyder, at kommaet nu skal flyttes 8 pladser *til venstre*, så vi får altså i virkeligheden 0.0000000504123818. Din terning viste 0.0000000504123818 øjne!

Men når nu RND(1) giver et tal mellem 0 og 1, må RND(1)\*6 da give et tal mellem 0 og 6. Nemlig, men noget helt tal er det stadig væk ikke. Det kan vi imidlertid lave det om til med en anden funktion:

INT(A)

betyder nemlig: Lav A om til et helt tal (smid decimalerne væk)!

INT(27.2345)

er 27. Og eftersom det tilfældige tal mellem 0 og 1, som maskinen skaber, aldrig er *helt* 1, giver

?INT(RND(1)\*6)

os et tilfældigt helt tal mellem 0 og 6, og derfor kan vi bruge

?INT(RND(1)\*6)+1

til at efterligne en terning. Et »endnu mere tilfældigt« tal kan vi få med

RND(0)

Denne version laver sit tilfældige tal efter et lille ur inde i maskinen, og det er derfor temmelig tilfældigt. Bad vi om det 10 eller 10 1/4 sekund over 5? Faktisk »går« maskinen med 1/60 sekunds nøjagtighed. Vi kan faktisk også bruge VIC som ur. Funktionen

TI

giver os antallet af tresindstyvendede dele af et sekund, der er gået, siden vi tændte for maskinen.

Bedre ville det selvfølgelig være, hvis vi kunne få tiden i timer, minutter og sekunder, og det kan vi faktisk med

TI\$

Prøv

```
1 PRINT"h"TI$
2 RUN
```

Eller du kan »stille uret«. Er klokken 12.34.56, indtaster du blot

```
TI$="123456"
```

eller du kan nulstille det

```
TI$="000000"
```

og bruge det som stopur i spil med mere. Eller måske kunne du have lyst til at vide, hvor meget hukommelse du har i reserve. Tag

```
?FRE(0)
```

Læg mærke til, at vi også kunne sige

```
?FRE(2345)
```

Her er tallet i parentesen ligegyldigt og er kun med for syntaksens skyld.

Der er selvfølgelig mange andre funktioner i VIC, som vi skal udforske efterhånden. I denne lektion har vi kun opdaget netop så meget, at vi kan lave et spilleprogram med lidt mening i. Vi laver hele to i næste lektion.

NB! Prøv

```
1 PRINT"FIND MIT TAL (0-9)":A=INT(RND(1)*10)
2 INPUTB:C=C+1:IFB=ATHENPRINT"DU FIK DET
  I"C"FORSØG":END
3 PRINT"NEJ - PRØV IGEN":GOTO2
```

# LEKTION 9

## RELATIONER

For at forstå denne lektions to programmer er der endnu et fænomen, vi skal kende til: RELATIONEN.

Relationerne er

= Lig med

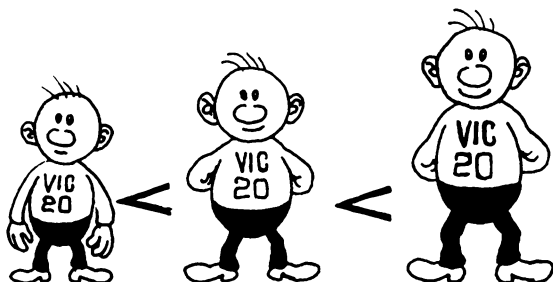
< > ULIG, FORSKELLIG FRA

> STØRRE END

< MINDRE END

>= STØRRE END ELLER LIG MED

<= MINDRE END ELLER LIG MED



Her kommer så det første program:

```
1 A=1000:B=10000
2 IFA=0THENPRINT"DU ER FALLIT":END
3 IFB <=0THENPRINT"BANKEN ER FALLIT":END
4 PRINT"KAPITAL:"A:PRINT"cOcOcOcOcOcOcOc
  OcO":PRINT"f11f92f0f33f94f0f55f96f0f67f98
  f0f79":PRINT"cUcUcUcUcUcUcUcUcU"
```

```

5 INPUT" TAL";C:INPUT" INDSATS";D:
  IFD > A THEN 5
6 A=A-D:B=B+D:E=INT(RND(1)*10):
  PRINT"f3"E"KOM UD"
7 IFE=C THEN PRINT"DU VANDT":A=A+D*10:
  B=B-D*10:GOTO 2
8 PRINT"f6DU TABTE":GOTO 2

```

Linjer som 4 med mange farvekontroller skal man lige vænne sig til at læse, men så skulle der heller ikke være noget at tage fejl af: Farve 1: 1, farve 9 (RVS ON): 2, farve 0 (RVS OFF), farve 3: 3, osv. (lige tal er omvendte). Lad os igen prøve at »oversætte«:

1. Du skal have 1000 kr., og banken skal have 10000.
2. Hvis du har mistet alle dine penge, er du fallit, og spillet slutter.
3. Hvis banken har mistet alle sine penge, er den fallit, og spillet slutter.
4. Her er alt det, der skal stå på skærmen: Hvor mange penge, du har tilbage, de 9 tal, der kan spilles på.
5. Nu kan du indtaste, hvilket tal du holder på, og hvor meget. Hvis det er mere, end du har, bliver du spurgt en gang til.
6. Din indsats bliver trukket fra din kapital, banken får den, rouletten snurrer, og resultatet printes ud.
7. Hvis du har vundet, får du det at vide, du får dine penge ti gange igen, og et tilsvarende beløb bliver trukket fra bankens kapital, og du kan prøve en gang til.
8. Hvis du har tabt, får du det at vide, og du kan prøve en gang til.

Det var da ikke så svært, vel? Prøv selv at oversætte det følgende eksempel. m betyder mellemrum. Sæt selv farver til programmet.

```

1 PRINT"VI LEGER FIND TALLET. DIT TAL ER
  MELLEM 1 OG":INPUTA:B=A:PRINT"OKAY,"A
2 PRINT"HVIS DIT TAL ERmmmmmmSTØRRE,
  TAST S; HVISmmMINDRE, M; HVIS DET ER
  RIGTIGT, R"
3 C=1:D=INT((B-C)/2)+1
4 E=E+1:PRINTE". FORSØG:"D:IFB=CTHEN10
5 INPUTA$:IFA$="R"THEN10
6 IFA$="S"THENC=D+1:D=D+INT((B-D)/2)
7 IFA$="M"THENB=D-1:D=C+INT((D-C)/2)
8 IFB-D=1THEND=D+1
9 GOTO4
10 PRINT"JEG FANDT"D:PRINT"T"E"FORSØG."
11 PRINT"KAN DU GØRE DET BEDRE?MIT TAL
  ER MELLEM 1 OG"A
12 PRINT"OPGIVER DU, KAN DUmMMM TASTE 0":
  B=INT(RND(1)*A)+1
13 INPUTA:IFA=0THEN23
14 F=F+1:PRINTF". FORSØG:"A:IFA=BTHEN18
15 IFB > ATHENPRINT"S"
16 IFB < ATHENPRINT"M"
17 GOTO13
18 PRINT"DU FANDT MIT TAL":PRINT"T"F
  "FORSØG.":PRINT"JEG FANDT DIT I"E"."
19 IFE < FTHENPRINT"JEG VANDT"
20 IFE > FTHENPRINT"DU VANDT"
21 IFE=FTHENPRINT"UAFGJORT"
22 END
23 PRINT"OKAY, DET VAR"B:PRINT"JEG VANDT"

```

Vi er nu tæt ved at være igennem begyndelsesgrundene i BASIC. Vi ved, hvad en variabel og et program er, kan bruge tastaturet og skrive primitive programmer med INPUT, IF og enkelte funktioner. I næste lektion skal vi fuldende vore forkundskaber med de logiske operationer AND, OR

og NOT. Her vil vi benytte lejligheden til at samle nogle småting op.

Vi brugte en del mellemrum i det sidste program. Faktisk kan sådanne overflødiggøres med funktionen

SPC(A)

hvor A er et tal. Prøv

?”VIC”20

og

?”VIC”SPC(7)20

eller

?SPC(7)”VIC”

I det sidste eksempel fik vi en lille »margen« på 7 pladser, næsten ligesom tabulatoren på en skrivemaskine. Faktisk har VIC en tabulatorfunktion, og

?TAB(7)”VIC”

vil give det samme resultat som ovenstående.

Vi slutter af med et par FEJLMEDDELELSER, som du måske er stødt ind i. Prøv

```
1 INPUTA:IFA<0THEN3
2 PRINTSQR(A):END
4 PRINT”DUER IKKE”
```

Tast 4. Du får kvadratroden af 4, som er 2. Indtast nu -4. Man kan ikke tage kvadratroden af et negativt tal, derfor

har vi også en linje 4, der siger "DUER IKKE". Men 1 sender os til 3, som ikke eksisterer, så vi får

```
?UNDEF'D STATEMENT  
ERROR IN 1
```

UNDEFINED STATEMENT betyder udefineret ordre, det vil sige: der er ingen mening i, hvad du siger! men datamater har jo altid en høfligere måde at sige den slags sandheder på.

Hvad gør vi nu? Jo, vi kører naturligvis op og retter 3-tallet i linje 1 til et 4-tal. Så skulle maskinen faktisk kunne fortsætte:

```
CONT
```

Den svarer

```
?CAN'T CONTINUE  
ERROR
```

Det er jo et helt andet program, altså kan den ikke »fortsætte«!

NB! Prøv

```
1 PRINT"JEG FINDER DIT TALmmmm(<=100),  
  HVIS DU GØR, SOM JEG SIGER. DEL DET"  
2 INPUT"MED 3! REST";A:INPUT"MED 5! REST"  
  ;B:INPUT"MED 7! REST";C:D=A*70+B*21+C*15  
3 IFD > 105THEND=D-105:GOTO3  
4 PRINT"DIT TAL ER"D
```



# LEKTION 10

## LOGISKE OPERATIONER

Prøv

```
1 INPUT"STØRSTE PLANET";A$:IFA$="JUPITER"  
  THENEND  
2 RUN
```

Programmet stiller opgaven, til det rigtige svar indtastes.  
Prøv nu

```
1 INPUT"MARS' DRABANTER";A$,B$:IFA$="PHO  
  BOS"ANDB$="DEIMOS"THENEND  
2 RUN
```

Nu standser programmet kun, hvis *begge* svar er rigtige.  
Men sæt nu, eleven svarer i omvendt rækkefølge? Så må vi  
have

```
1 INPUT"MARS' DRABANTER";A$,B$  
2 IFA$="PHOBOS"ANDB$="DEIMOS"ORA$=  
  "DEIMOS"ANDB$="PHOBOS"THENEND  
3 RUN
```

OR betyder eller. Hvis blot en af betingelserne er opfyldt,  
standser programmet, altså enten

1. A\$="PHOBOS" og B\$="DEIMOS

eller

2. A\$="DEIMOS" og B\$="PHOBOS"

Men hvordan kan maskinen nu vide, at vi ikke mener, at programmet skal stoppe, hvis det er opfyldt, at

1. A\$="PHOBOS"

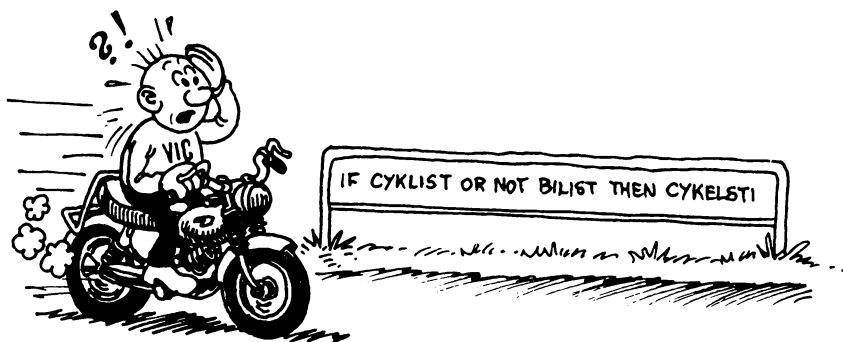
og

2. B\$="DEIMOS" eller A\$="DEIMOS"

og

3. B\$="PHOBOS"

Dette ville jo aldrig ske, for hvis A\$ og B\$ begge er "PHOBOS", kan ingen af dem være "DEIMOS". Svaret er igen »prioritering«. Maskinen tager simpelt hen alle AND, før den begynder med OR. Der er imidlertid en logisk operation, den udfører før både AND og OR, og det er NOT: ikke.



Lad os igen se på lektionens første eksempel. Faktisk kunne vi have nøjedes med en enkelt linje:

```
1 INPUT"STØRSTE PLANET";A$:IFNOTA$="JUPITER"THENRUN
```

Nu siger programmet i stedet for, at maskinen skal spørge igen, hvis svaret *ikke* er rigtigt, og det kommer jo faktisk ud på et. Kunne vi gøre det samme med det næste eksempel:

```
1 INPUT"MARS' DRABANTER";A$,B$:IFNOT(A$="PHOBOS"ANDB$="DEIMOS")THENRUN
```

Hvad skal vi med parenteser? Ja, hvad ville

```
1 INPUT"MARS' DRABANTER";A$,B$:IFNOTA$="PHOBOS"ANDB$="DEIMOS"THENRUN
```

betyde? Nemlig: Spørg igen, hvis det er opfyldt, at

1. A\$ ikke er "PHOBOS"

og

2. B\$ er "DEIMOS"

Maskinen tager NOT før AND, ikke? Måske synes du, linjen ser lidt mystisk ud. Kunne vi ikke omskrive den på en måde, så vi kan undvære parenteser? Hvad med

```
1 INPUT"MARS' DRABANTER";A$,B$:IFNOTA$="PHOBOS"ANDNOTB$="DEIMOS"THENRUN
```

Det ser da plausibelt ud, ikke? Men pas nu på: Det betyder jo, at *både* PHOBOS og DEIMOS skal være forkerte, hvis ma-

skinen skal gå tilbage. Og da vi jo må forlange begge rigtige, for at maskinen skal stoppe, skal også den ene forkert være nok til at sende maskinen tilbage.

```
1 INPUT"MAR$' DRABANTER";A$,B$:IFNOT(A$="P
  HOBOS"ANDB$="DEIMOS")THENRUN
```

må altså oversættes ved

```
1 INPUT"MAR$' DRABANTER";A$,B$:IFNOTA$="PH
  OBOS"ORB$="DEIMOS"THENRUN
```

Hvad nu, hvis linjen havde heddet

```
1 INPUT"MAR$' DRABANTER";A$,B$:IFNOT(A$="P
  HOBOS"ORB$="DEIMOS")THENRUN
```

Hvornår ville maskinen så være gået tilbage? *Når det ikke var tilfældet, at*

$(A\$="PHOBOS"ORB\$="DEIMOS")$ . Med andre ord, når *ingen af delene* var tilfældet. Dette udtrykker altså i virkeligheden et »hverken-eller« og kan oversættes

```
1 INPUT"MAR$' DRABANTER";A$,B$:IFNOTA$="PH
  OBOS"ANDNOTB$="DEIMOS"THENRUN
```

Vi kan også udtrykke os lidt mere generelt, idet vi lader bogstaverne P og Q (en slags variabler) stå for hele relationer. Så kan vi skrive

$\text{NOT}(\text{PANDQ})$  er det samme som  $\text{NOTPORN} \text{NOTQ}$

NOT(PORQ) er det samme som NOTPANDNOTQ

Husk endelig på, at

NOTA=B

er præcis det samme som

$A < > B$

På samme måde er  $A > B$  det modsatte af  $A \leq B$  og  $A < B$  det modsatte af  $A \geq B$ .

Vi er nu som sagt gennem begyndelsesgrundene i BASIC. I næste del skal vi udforske sprogets muligheder nærmere.

NB! Prøv

```
1 A=25:PRINT" NIM"
2 PRINT"DER ER" A "TILBAGE":IFA < 2THENPRINT"
  JEG VANDT":END
3 INPUT"HVOR MANGE TAGER DU";B:IFB
  < 1ORB > 3THEN3
4 C=4-B:PRINT"JEG TAGER"C:A=A-B-C:GOTO2
```



*ANDEN DEL*  
*Data og sæt*





# LEKTION 11

## FOR-NEXT-LØKKER

Prøv

```
10 PRINT"VIC 20"  
20 GOTO10
```

Som vi tidligere har set, er dette en løkke, der skriver VIC 20 på skærmen, til vi taster RUN STOP. Det er en såkaldt »uendelig« og komplet ukontrollabel løkke. Vi kan imidlertid kontrollere den med

```
10 PRINT"VIC 20"  
20 A=A+1:IF A < 10THEN10
```

Dette kan vi imidlertid også udtrykke på en anden måde:

```
5 FORA=1TO10  
10 PRINT"VIC 20"  
20 NEXTA
```

Fænomenet kaldes en FOR-NEXT-LØKKE. A er en KONTROLVARIABLE. Den svarer til tælleren i det første eksempel, idet den gennemløber de værdier, FOR-linjen definerer, og således sørger for, at alt mellem FOR og NEXT bliver udført et bestemt antal gange. Vi kan sagtens PRINTe kontrolvariablen ud, ligesom vi kan have flere FOR-NEXT-løkker inde i hinanden:

```
1 FORA=1TO10:FORB=1TO10:PRINTA*B:NEXTB:NEXTA
```

skriver den lille tabel ud. Læg mærke til, at løkkerne skal ligge inden i hinanden, så maskinen ikke støder på en »forkert« NEXT. Et andet legalt eksempel ville være

```
FORA
FORB
FORC
NEXTC
NEXTB
FORD
NEXTD
NEXTA
```



Ofte kan vi undvære at nævne kontrolvariablen efter NEXT. Så vil maskinen nemlig automatisk opfatte det som hørende til den sidst påbegyndte løkke.

Prøv

```
1 FORA=1TO10:PRINT"VIC 20"
```

og

```
1 PRINT"VIC 20":NEXT
```

Prøv nu

```
1 FORA=10TO0STEP.5:PRINTA:NEXT
```

Nu tæller kontrolvariablen i trin af .5 (1/2). Vi kan også have negativt STEP:

```
1 FORA=10TO0STEP-1:PRINTA:NEXT
```

Kontrolvariablen giver os faktisk en ret enestående kontrol. Tag f.eks. vores tabelprogram, hvor tallene bare farer af sted. Vi kunne nu have

```
1 FORA=1TO10:FORB=1TO10:PRINTA*B:NEXT:INP  
UT:NEXT
```

eller

```
1 FORA=1TO10:FORB=1TO10:PRINTA*B:NEXT:FO  
RC=1TO5000:NEXT:NEXT
```

Læg mærke til, at C-løkken ingenting gør. Den er bare en pause. Hvor lang er den? Prøv

```
1 TI$="000000":FORA=1TO10000:NEXT:PRINTTI/60
```

Maskinen »tæller til« 10000 på 9.38333334 sekunder. En datamats kvalitet kan ofte ses på dens hurtighed, og VIC 20 er faktisk temmelig hurtig i forhold til sine konkurrenter, TEXAS, COLOR GENIE, SPECTRUM osv. Allerlangsomst (men også allerbilligst) er selvfølgelig ZX81, der ville være 3 minutter og 20 sekunder om ovenstående løkke. SPECTRUM ville kunne klare den på 50 sekunder.

Lidt forenklet kan vi sige, at skal vi have en pause på 1 sekund, skal FOR-NEXT-løkken på VIC gå til 1000. Det kan måske synes lidt overflødigt med en så stor hurtighed i udregningsprogrammer, men vi bliver glade for den, når vi skal til at lave TV-spil!

Eller prøv

```
1 FORA=1TO20:FORB=1TO22:PRINT"f3S";:NEXT:NE  
XT
```

Vi får 20 rækker hjerter. Prøv at skifte As højeste værdi (dens »grænseværdi« – 1 er »startværdien«) ud med 21. Vi fik ikke nogen ekstra linje, tværtimod.

Maskinen skaffer plads til sit READY.

Prøv nu

```

1 PRINT"f3":FORA=1TO23:FORB=1TO22:IFA=23AND
  DB=22THENPRINT"vi";
2 PRINT"S";:NEXT:NEXT
3 GOTO3

```

i står for INSERT. Når du kommer til denne karakter, holder du bare SHIFT nede og taster INST. Kan du se virkningen? Maskinen skifter linje, når vi har skrevet på dens sidste plads. Men i dette program skriver vi *aldrig* på den sidste plads, vi skriver på den *næstsiste*, og så går vi et skridt tilbage (*v = markør til venstre*) og laver et hul (INSERT), før vi skriver den sidste karakter. Endelig »fryser« vi billedet med 3. 2 kunne også have heddet

```

2 PRINT"S";:NEXTB,A

```

De to linjer er ens, kun udtryksmåden er forskellig. Læg også mærke til, at de mange 1-linjes-eksempler sagtens kunne undvære linjenummer og indtastes som direkte ordrer, selv om de i virkeligheden er sammensat af flere sådanne! Prøv til slut

```

1 INPUT"PRIMTAL MELLEM";A:INPUT"OG";B:
  IFA <=2THENA=2:PRINT2;:C=1
2 FORD=ATO B:FORE=2TOSQR(D):IFD/E=INT(D/E)
  THEN4
3 NEXTE:PRINTD;:C=C+1
4 NEXTD:PRINTC"PRIMTAL"

```

NB! Prøv

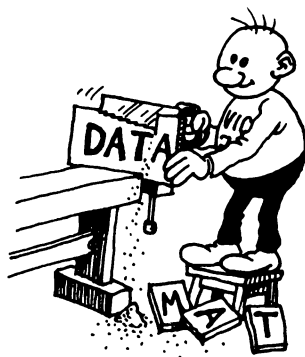
```

1 A=1:PRINT"s3f9MATEMATIKs": INPUTB: FORC
  =1TOB:A=A*C:NEXT:PRINTB"v!(FAKULTET)=":PR
  INTA:RUN

```

## LEKTION 12

### STRENGDELING



Kan du huske

- 1 INPUT"MAR'S' DRABANTER";A\$,B\$
- 2 IFA\$="PHOBOS"ANDB\$="DEIMOS"ORA\$="DEIMOS"ANDB\$="PHOBOS"THENEND
- 3 RUN

Den lange linje skyldtes jo, at vi ikke kunne vide, om eleven ville svare PHOBOS eller DEIMOS *først*. Men tag så f.eks.

- 1 INPUT"STØRSTE FUGL";A\$:IFA\$ < > "STRUDS" THENRUN

Sæt nu, der svares EN STRUDS eller STRUDSEN eller STRUDSE? Ja, så er svaret »forkert«. Det ville kræve alt for mange ANDer at dække alle mulige korrekte besvarelser. Problemet er jo, at maskinen ikke virkelig forstår, hvad vi siger.

Hvordan forstår vi da selv svar som ovenstående som rigtige? Jo, vi ser efter, om de indeholder STRUDS. Det kan maskinen også gøre. Til det brug har den nemlig en række funktioner:

LEN(A\$) betyder antallet af karakterer i strengen A\$  
LEFT\$(A\$,A) betyder de A første karakterer i strengen A\$  
RIGHT\$(A\$,A) betyder de A sidste karakterer i strengen A\$  
MID\$(A\$,A,B) betyder B karakterer fra strengen A\$ begyndende med karakter nummer A

I det sidste tilfælde kan vi underforstå B. Så betyder

MID\$(A\$,A) strengen A\$ fra og med karakter nummer A, og strengen ud.

Læg mærke til, at

MID\$(A\$,A,1)

åbenbart betyder: Karakter nummer A i strengen A\$! Lad os prøve funktionerne på strengen

A\$="KØBENHAVN"

Vi får nu f.eks.

LEN(A\$)=9

LEFT\$(A\$,3)="KØB"

RIGHT\$(A\$,4)="HAVN"

MID\$(A\$,4,2)="EN"

MID\$(A\$,4)="ENHAVN"

MID\$(A\$,4,1)="E"

Eller prøv

?LEFT\$(A\$,5)+RIGHT\$(A\$,3)

```
1 A$="KØBENHAVN":FORA=1TO8:PRINTMID$(A$,  
  A,2):NEXT
```

skriver alle mulige DELSTRENGE af A\$ på 2 karakterer ud.  
Laver vi dette program om til

```
1 A$="KØBENHAVN":FORA=1TO8:IFMID$(A$,A,2)=  
  "EN"THENPRINT"OK"  
2 NEXT
```

får vi et program, der checker, om en af disse delstreng er  
"EN". Vi kan nu lave vores strudse-program om til

```
1 INPUT"STØRSTE FUGL";A$:FORA=1TOLEN(A$):IF  
  MID$(A$,A,6)="STRUDS"THENEND  
2 NEXT:RUN
```

Vi kan også gå den anden vej og lave et program, der kan  
konstatere forekomsten af forskellige delstreng i strengen  
"KØBENHAVN". Prøv

```
1 A$="KØBENHAVN":INPUT"DELSTRENG";B$:FORA  
  =1TO9:IFMID$(A$,A,LEN(B$))=B$THENPRINT"OK"  
2 NEXT
```

eller helt generelt

```
1 INPUT"STRENG";A$:INPUT"DELSTRENG";B$  
2 FORA=1TOLEN(A$):IFMID$(A$,A,LEN(B$))=B$THEN  
  PRINT"OK"  
3 NEXT
```

Hvad gør

```
1 INPUTA$:FORA=1TOLEN(A$):IFMID$(A$,A,1)="A"  
  THENB=B+1  
2 NEXT:PRINTB
```

Eller prøv

```
1 INPUT"TEKST";A$:FORA=1TOLEN(A$)
2 IFMID$(A$,A,8)="COMPUTER"THEN A$=LEFT$(A$,
  A-1)+"DATAMAT"+RIGHT$(A$,LEN(A$)-A-7)
3 NEXT:PRINTA$
```

Programmet undersøger A\$ for forekomsten af delstren-  
gen COMPUTER og skifter den ud, hvor den forekommer,  
med det mere danske DATAMAT. Generelt kan vi have

```
1 INPUT"TEKST";A$:INPUT"RET";B$:INPUT"TIL";C$
  :FORA=1TOLEN(A$)
2 IFMID$(A$,A,LEN(B$))=B$THEN A$=LEFT$(A$,A-1)
  +C$+RIGHT$(A$,LEN(A$)-A-LEN(B$)+1)
3 NEXT:PRINTA$
```

Prøv

```
1 PRINTMID$("f1f2f3f4f5f6f7f8",INT(RND(1)*8)+1,1)"f9
  m";:RUN
```

og

```
1 INPUT"TEKST";A$:FORA=1TO22-LEN(A$):A$=A$+
  "m":NEXT:PRINT"h"
2 PRINT"hf7f9mmmmmmmmmmmmmmmmmmmmmmmm
  mmmf3"LEFT$(A$,22)"f7mmmmmmmmmmmmmmmm
  mmmmmmmmmmm"
3 A$=RIGHT$(A$,LEN(A$)-1)+LEFT$(A$,1):FORA=1T
  O100:NEXT:GOTO2
```

NB! Prøv til sidst

```
1 INPUT"ORD";A$:FORA=LEN(A$)TO1STEP-1: B$ =
  B$ + MID$(A$,A,1):NEXT:A$=B$:PRINTA$:RUN
```

72



# LEKTION 13

## GOSUB

Som vi har set, giver

```
1 A$="KØBENHAVN":PRINTMID$(A$,3,1)
```

os bogstavet B. Kunne vi ikke på samme måde bede om det tredje *ciffer* i f.eks. tallet 123456789? Vi prøver

```
1 A=123456789:PRINTMID$(A,3,1)
```

Vi får TYPE MISMATCH (se lektion 6), eftersom vi forsøger at bruge en funktion, der kun egner sig til strenge, på et tal. Det, vi skal bruge, er en funktion, der laver et tal om til en streng. Den har vi i

STR\$

Prøv

```
1 A=123456789:A$=STR$(A):PRINTMID$(A$,3,1)
```

Vi får 2! Ja, for strengen A\$ er nemlig på 10 karakterer, hvad vi kan overbevise os om med LEN(A\$). Den første karakter er et mellemrum, der i virkeligheden er et underforstået +, hvilket betyder, at A er et positivt tal. Sæt, vi nu vil have vores 2-tal ganget med 3? Vi forsøger

```
1 A=123456789:A$=STR$(A):PRINTMID$(A$,3,1)*3
```

Mere miskmask. Nu prøver vi gud hjælpe mig at gange en streng med 3. Nu skal vi bruge

VAL

der laver en streng om til et tal. Vi kan så have

```
1 A=123456789:A$=STR$(A):B=VAL(MID$(A$,3,1)):PR  
INTB*3
```

Prøv

```
1 A$="f1f2f3f4f5f6f7f8":FORA=1TO4:B=B+(INT(RN  
D(1)*8)+1)*10p(6-A):NEXT  
2 B$=MID$(STR$(B),2,4)  
3 C=C+1:PRINTC;:PRINT"v. FORSØG";:IFC < 10TH  
ENPRINT"ø";  
4 INPUTC$:PRINT"nøoooooooooooooooo";:FORA=1TO4:P  
RINTMID$(A$,VAL(MID$(C$,A,1)),1)"Q";:NEXT  
5 D=0:E=0:FORA=1TO4:IFMID$(C$,A,1)=MID$(B$,A,  
1)THEND=D+1:GOTO8  
6 FORB=1TO4:IFMID$(C$,A,1)=MID$(B$,B,1)THENE=  
E+1:GOTO8  
7 NEXTB  
8 NEXTA:PRINT"f7";:IFD=0THEN10  
9 FORA=1TOD:PRINT"*";:NEXT  
10 IFE=0THEN12  
11 FORA=1TOE:PRINT".";:NEXT  
12 IFC$=B$THENEND  
13 PRINT:GOTO3
```

Frit oversat:

1. Maskinen tænker på et tal på fire cifre mellem 1 og 8.

2. Ditto.
3. Maskinen regner ud, hvor mange forsøg du har brugt.
4. Maskinen tager imod dit gæt og skriver det på skærmen som farver. Tolket på denne måde er det en farvecombination, den har tænkt på, og som du skal gætte ved at indtaste tallene på farvetasterne. Er du med?
5. Maskinen undersøger, om du har placeret nogen farver korrekt.
6. Maskinen undersøger, om du har placeret en rigtig farve et forkert sted.
7. Slut på B-løkken.
8. Slut på A-løkken. Skift til »normal« farve. Hvis ingen farver er placeret korrekt, videre til 10!
9. Maskinen skriver, hvor mange farver korrekt placeret.
10. Hvis ingen rigtige farver placeret forkert, videre til 12!
11. Maskinen skriver, hvor mange rigtige farver forkert placeret.
12. Hvis alle rigtige, slut!
13. Ny linje. Om igen, soldat!

Prøv også

```

1 A$="HVAD ER HOVEDSTADEN I":PRINTA$:INPUT
  T"DANMARK";B$:C$="KØBENHAVN":GOSUB7
2 PRINTA$:INPUT"SVERIGE";B$:C$="STOCKHOLM":
  GOSUB7
3 PRINTA$:INPUT"NORGE";B$:C$="OSLO":GOSUB7
4 PRINTA$:INPUT"FRANKRIG";B$:C$="PARIS":GOS
  UB7
5 PRINTA$:INPUT"ITALIEN";B$:C$="ROM":GOSUB7
6 PRINTA$:INPUT"SPANIEN";B$:C$="MADRID":GOS
  UB7:END
7 FORA=1TOLEN(B$):IFMID$(B$,A,LEN(C$))=C$THEN
  PRINT"RIGTIGT":RETURN
8 NEXT:PRINT"NEJ - "C$:RETURN

```

Det er igen strengdelingen, der er på spil. Men i stedet for at sætte denne strengdeling efter hvert spørgsmål, findes den kun ét sted i programmet, nemlig linje 7 og 8. Det lille ord GOSUB sender ligesom GOTO maskinen til denne linje for hvert spørgsmål, men det særlige ved GOSUB er, at den selv finder »hjem« igen. RETURN betyder »gå videre i programmet, hvor du slap«.

Læg mærke til END i linje 6. Prøv at slette det. Vi får

```
?RETURN WITHOUT GOSUB  
ERROR IN 7
```

eller måske (hvis du svarer forkert) ERROR IN 8. Jo, se, da maskinen havde klaret det sidste spørgsmål, fortsatte den i 7 og stødte derfor på et RETURN uden GOSUB. Den var jo nemlig ikke blevet sendt ind i denne SUBROUTINE af et GOSUB, men var af vanvare vadet derind »efter kampen«.

Læg også mærke til, at vi ikke behøver at skrive HVAD ER HOVEDSTADEN I for hvert spørgsmål, eftersom vi har oplagret denne (faste) del af spørgsmålet i A\$.

Selvfølgelig skal vi stadig bruge en lang linje til hvert enkelt spørgsmål. Tænk, hvis vi i stedet kunne proppe det hele ind i en enkelt FOR-NEXT-løkke: Spørgsmål 1, svar 1, osv. Det kan vi faktisk godt. Men så skal vi først vide noget om SÆT!

NB! Prøv

```
1 INPUT "TAL";A:A$=STR$(A):FORA=2TOLN(A$):  
  B=B+VAL(MID$(A$,A,1)):NEXT:PRINTB:RUN
```

# LEKTION 14

## SÆT

Et sæt er et system af variabler, hvis navne kun adskiller sig numerisk fra hinanden, hvilket giver os mulighed for at behandle dem helt anderledes effektivt.

```
1 FORA=1TO10:A(A)=37:NEXT
```

tilskriver på en og samme gang værdien 37 til ti forskellige variabler, som vi senere kan bruge som A(1), A(2), . . . A(10). Men prøv så at køre

```
1 FORA=1TO100:A(A)=37:NEXT
```

Vi får

```
?BAD SUBSCRIPT  
ERROR IN 1
```

Fejlmeddelelsen betyder »forkert indeks«. Indeks er tallet i parentes. Det er klart, at vi på én gang forlanger en forfærdelig masse plads af maskinen, når vi i en enkelt linje vil tilskrive 100 variabler. Skal vi derfor danne sæt på over 11 INDICEREDE VARIABLER, må vi først gøre maskinen opmærksom på, at vi kommer til at behøve denne plads. Det gør vi med ordren

```
DIM
```

Vi kan så have

```
1 DIMA(100):FORA=1TO100:A(A)=37:NEXT
```

og checke med

```
?A(77)
```

Det fungerer. Faktisk har vi med DIMENSIONERINGS-ORDREN skaffet plads til 101 variabler, idet den første variabel i et sæt altid har indeks 0! Vi kan også dimensionere STRENG-SÆT. Vi kunne altså sagtens have haft

```
1 DIMA$(100):FORA=1TO100:A$(A)="KØBENHAVN":  
NEXT
```

Endelig kan vi danne sæt i flere dimensioner. Prøv

```
DIMB(15,15)
```

Vi har nu  $16 \cdot 16 = 256$  variabler til rådighed med navnene  $B(0,0), B(0,1), B(0,2), \dots B(0,15), B(1,0), \dots B(15,15)$

Prøv nu

```
DIMB(15)
```

Vi får

```
?REDIM'D ARRAY  
ERROR
```

På denne måde forhindrer maskinen os i at slette et gammelt sæt med et nyt ved en fejltagelse.

En mand ved navn Conway opstillede engang et sæt (temmelig forenklede) leveregler for levende celler. Han gik

ud fra en »flad« koloni, som den kunne brede sig over en glasplade, altså

```
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
```

Som det vil fremgå, har hver celle, bortset fra de yderste, 8 naboer – dette er selvfølgelig efter dette system det største antal naboer, en celle kan have. I kolonien

```
0
00
```

har hver celle 2 naboer. CONWAYS REGLER siger nu:

1. En celle med 2 eller 3 naboer overlever til næste generation.
2. En celle »fødes« i *næste generation* i et felt, der er tomt i *denne generation*, hvis dette tomme felt har nøjagtig 3 naboer.

Underforstået i 1. er, at en celle med *under 2* eller *over 3* naboer dør. Dette skal vi nu forsøge at lave om til et program: LIFE. Vi prøver

```
1 DIMA%(23,24),B%(23,24):A=-1:FORB=2TO22:FORC
  =2TO23:IFRND(1)<.5THENB%(B,C)=1
```

```

2 NEXT:NEXT:PRINT"h"
3 A=A+1:PRINT"h4f9GENERATION"A"f3":D=0:FOR
  B=2TO22:FORC=2TO23:A%(B,C)=B%(B,C)
4 IFA%(B,C)=1THENPRINT"Q";:D=D+1
5 IFA%(B,C)=0THENPRINT"m";
6 NEXT:NEXT:PRINT"f4f9POPULATION"D"vm";:FO
  RB=2TO22:FORC=2TO23:D=0:D=D+A%(B-1,C-1)
7 D=D+A%(B-1,C)+A%(B-1,C+1)+A%(B,C-1)+A%(B,
  C+1)+A%(B+1,C-1)+A%(B+1,C)+A%(B+1,C+1)
8 IFA%(B,C)=1ANDD < > 2ANDD < > 3THENB%(B,C)
  =0
9 IFA%(B,C)=0ANDD=3THENB%(B,C)=1
10 NEXT:NEXT:GOTO3

```

Og så er vi parate til en lille studie i cellevækst. Jeg startede selv med 225 celler, men det var åbenbart en temmelig kraftig overbefolkning (prøv selv at pille ved linje 1!). I løbet af fem generationer var jeg nede på en population på 66, men ved 7. generation (54), begyndte det igen at gå lidt op ad bakke, og i de næste ti generationer stabiliserede populationen sig omkring de tres. Jeg bliver aldrig træt af dette simple program; at sidde og betragte de besynderlige livsformer, der udvikler sig spontant, udelukkende for at overleve under de givne naturlove. Hvordan de indvirker på hinanden og endelig stabiliseres i perfekte former eller dør.

Ved 25. generation havde jeg 78 celler, ved 50. 60. Fire »organismer« var på dette tidspunkt fuldkommen stabile, hver med sin særprægede form. Men prøv nu selv!

**NB! Prøv**

```

1 INPUT"ANTAL TERNINGER";A:INPUT"ANTAL
  KAST";B:DIMA(A*6)
2 FORC=1TOB:FORD=1TOA:E=E+INT(RND(1)*6)+1:
  NEXT:A(E)=A(E)+1:E=0:NEXT
3 FORC=1TOA*6:PRINTC": "A(C) *100/B "%":NEXT

```



# LEKTION 15

## DATA

I lektion 14 brugte vi et NUMERISK SÆT I 2 DIMENSIONER til at fremstille et program, der simulerede en cellekolonis liv og død. I denne lektion skal vi se lidt på, hvad vi kan bruge STRENG-SÆT til. Prøv

```
1 DIMA$(20):FORA=1TO20:READA$(A):NEXT
2 INPUTA$:FORA=1TO20:FORB=1TOLEN(A$(A)):IF
   MID$(A$(A),B,LEN(A$))=A$THENPRINTA$(A)
3 NEXT:NEXT:GOTO2
4 DATAKLAUS RIFBJERG*JUS OG/ELLER DE
   N GYLDNE MIDDELVEJ,JENS PEDERSEN*UDEN
   FOR MIDT I
5 DATAEGON NIELSEN*SALAMANDERDAGE,AN
   DERS BODELSEN*OVER REGNBUE
6 DATAERIK WEDERSØE*18 TONS DRØMME,DE
   A TRIER MØRCH*AFTENSTJERNEN
7 DATADORRIT WILLUMSEN*NI LIV,FINN ABRA
   HAMOWITZ*SKYGGEN FØR MØRKET
8 DATAVAGN LUNDBYE*JONAS TRILOGIEN,CH
   ARLOTTE STRANDGAARD*LILLE MENNESKE
9 DATAANNA LADEGAARD*REJSE MED MIN SØ
   N,ANGELO HJORT*DEN SOVENDE BY
10 DATAHANS LYNGBY JEPSEN*SOMMER I SEPT
   EMBER,LENE H. BAGGER*IDIOTERNE
11 DATAOLE FRØSTRUP*YETIERNES SANG, POUL
   HENRIK TRAMPE*FORHOLD
12 DATAERWIN NEUTZSKY*WULFF*MENNESKE,FL
   EMMING NORDENHOF*UDBRUD,HARLY FOGED
   *SKABET
```

## 13 DATAKNUD H. THOMSEN\*RØVERNE I SKOTLAND

Når vi kører dette program, viser det os et spørgsmålstegn: Signal til INPUT. Vi kan nu indtaste et SØGEORD, og programmet vil ud fra dette finde den rigtige bog. FOGED vil altså føre os frem til HARLY FOGED\*SKABET, og JONAS til Vagn Lundbye. Dette er ærketypen på en database, et datakartotek, der er alle kartoteker på én gang; efter forfatter, titel, eller hvad vi måtte ønske. Men hvordan virker det? Vi kan umiddelbart inddele ovenstående program i tre underprogrammer.

1. Initialisering.

2.-3. Analyse.

4.-13. Data.

Analysen er selvfølgelig programmets egentlige mekanisme, og den er faktisk ret banal strengdeling. Men vi skal jo også have vore oplysninger ind i sættet, og det er denne proces, vi hentyder til som initialisering. Vi kunne have haft

```
1 DIMA$(20):A$(1)="KLAUS RIFBJERG*JUS OG/ELL  
ER DEN GYLDNE MIDDELVEJ"
```

osv., men det havde ikke været særlig praktisk. En måde er

```
1 DIMA$(20):FORA=1TO20:INPUT$(A):NEXT
```

Nu »beder« maskinen simpelt hen om de 20 indtastninger og gemmer dem i sættet. Programmet er selvfølgelig lettere at overse, hvis sættets værdier står at læse i det, og jeg har derfor her valgt en tredje metode, en serie DATA-sætninger uden for det egentlige program. Formlen

READA\$(A)

betyder nu, at maskinen skal læse DATA ind i sættet et for et. Til terminologien hører også RESTORE, der betyder, at maskinen skal begynde forfra på DATA.

Tag

```
1 DIMA(100):FORA=1TO100:READA(A):IFA/7=INT(A/  
7)THENRESTORE  
2 NEXT  
3 FORA=1TO100:PRINTA(A);:NEXT  
4 DATA1,2,3,4,5,6,7,8,9,10
```

Læg mærke til, at vi ikke behøver gåseøjne om DATA, medmindre udeladelsen af dem kan give anledning til misforståelser, som hvis vi havde haft et , med i en af bogtitlerne. Faktisk brugte vi \* i stedet for : mellem forfatter og titel, eftersom : uden "" ville have betydet noget i retning af »slut på data«.

Prøv på denne måde at skære data fra i det sidste program, f.eks. efter 5. Maskinen klager

```
?OUT OF DATA  
ERROR IN 1
```

Metoden overflødiggør som sagt alfabetiske kartoteker, men ønsker vi netop alfabetisering, kan vi også få det:

```
1 DIMA$(100):A$(1)="ZZ"  
2 A=A+1  
3 INPUTA$:IFA$=":"THEN7  
4 FORB=1TO100:IFA$ < A$(B)THEN6  
5 NEXT  
6 FORC=ATOBSTEP-1:A$(C+1)=A$(C):  
NEXT:A$(B)=A$:GOTO2
```

7 FORB=1TOA-1:PRINTA\$(B):NEXT:GOTO3

Når du ønsker at se den (foreløbige) alfabetiserede liste, maskinen har lavet ud af dine indtastninger, skal du blot taste \*. Så kommer listen frem, hvorpå maskinen er parat til at modtage yderligere indtastninger.

Princippet i det hele er maskinens behagelige evne til at vurdere strenge ved at lade dem indgå i relationer.

A\$ < B\$

betyder altså simpelt hen: A\$ kommer før B\$ i alfabetet. Hvert bogstav har nemlig en numerisk værdi, den såkaldte ASCII-kode, som vi senere skal vende tilbage til.

Læg også mærke til, hvordan programmet hele tiden laver et »hul« til det nye ord ved at skubbe alle de tidligere en tak!

I 16. lektion skal vi bl.a. se på et program, der er i stand til at oversætte en given tekst fra et sprog til et andet!

NB! Prøv

```
1 A=A+1:READA$(A):IFA$(A) < > "SLUT"THEN1
2 FORA=1TO9:PRINTA,A$(A):NEXT
3 DATAET,TO,TRE,FIRE,FEM,SEKS,SYV,OTTE,NI,S
  LUT
```

# LEKTION 16

## STRENGSÆT

- 1 DIMA\$(86),B\$(86),C\$(25):FORA=1TO86:READA\$(A),  
B\$(A):NEXT:INPUTA\$:A\$=A\$+" "":A=2:B=1:C=1
- 2 IFMID\$(A\$,A,1)=" "THENC\$(B)=MID\$(A\$,C,A-C):B  
=B+1:C=A+1
- 3 IFA=LEN(A\$)THEN5
- 4 A=A+1:GOTO2
- 5 FORA=1TOB-1:FORC=1TO86:IFC\$(A)=A\$(C)THEN  
C\$(A)=B\$(C):GOTO7
- 6 NEXTC
- 7 PRINTC\$(A)" "":NEXTA
- 8 DATACOME,KOMME,GET,SKAFFE,GO,REJSE,KEE  
P,HOLDE,LET,LADE,MAKE,LAVE,PUT,ANBRINGE
- 9 DATASEEM,FOREKOMME,TAKE,TAGE,BE,BLIVE,  
DO,GØRE,SAY,SIGE,SEE,SE,SEND,SENDE,MAY,KA  
N
- 10 DATAWILL,VIL,ABOUT,OMKRING,ACROSS,OVE  
R,AFTER,EFTER,AGAINST,MOD,AMONG,IBLAND  
T,AT,VED
- 11 DATABEFORE,FØR,BETWEEN,IMELLEM,BY,AF,D  
OWN,NED,FROM,FRA,IN,I,OFF,BORT,THROUGH,  
IGENNEM
- 12 DATATO,TIL,UP,OP,WITH,MED,AS,LIGESOM,OF,  
AF,TILL,TIL,THAN,END,A,EN,THE,DEN,ALL,AL
- 13 DATAANY,NOGEN,EVERY,ENHVER,NO,NEJ,OTH  
ER,ANDEN,SOME,NOGEN,THAT,DEN,THIS,DENN  
E,I,JEG
- 14 DATAHE,HAN,YOU,DU,WHO,HVEM,AND,OG,BE  
CAUSE,FORDI,BUT,MEN,OR,ELLER,IF,HVIS
- 15 DATATHOUGH,SKØNT,WHILE,MEDENS,HOW,H

VORDAN, WHEN, DA, WHERE, HVOR, WHY, HVOR  
FOR, AGAIN, IGEN

16 DATAFAR, FJERN, FORWARD, FREMAD, HERE, HE  
R, NOW, NU, OUT, UD, STILL, ENDNU, THEN, DA, THE  
RE, DER

17 DATATOGETHER, SAMMEN, WELL, GODT, ENOUGH,  
NOK, EVEN, ENDOG, LITTLE, LILLE, MUCH, MEGET,  
N, NOT, IKKE

18 DATAONLY, KUN, QUITE, HELT, VERY, MEGET, NO  
RTH, NORD, SOUTH, SYD, EAST, ØST, WEST, VEST, YES,  
JA

Prøv nu programmet med originale og intelligente sætninger som

IF YOU GO WEST THEN I WILL COME WITH  
YOU

eller

I WILL NOT LET YOU GO THERE

Måske synes du ikke umiddelbart, at

HVIS DU REJSE VEST DA JEG VIL KOMME MED  
DU

eller

JEG VIL IKKE LADE DU REJSE DER

er synderlig gode oversættelser, men tænk så på, hvis der var tale om en russisk tekst, som du ikke forstod et eneste ord af. Så ville en oversættelse af denne type ikke være så ringe. Selvfølgelig er programmet ikke meget værd i den

form, det har nu. 86 glosers er ikke meget, og ikke et eneste navneord! Men vi kan da også sagtens få plads til flere. De 86 glosers med oversættelse fylder knap en K. 300 glosers skulle altså være inden for mulighedernes grænse. Med en 16K-udvidelse kommer vi op på to tusind, og så skulle vi have alle muligheder for at få et program, der fungerer. Bemærk dog, at vi så ikke som i dette program starter med at lade READ-linjen læse alle glosers ind fra DATA-linjerne, men tager en ind ad gangen, sammenligner, »smider den ud igen«, tager den næste ind, og så fremdeles. Lidt grammatik kunne man måske også indlægge i programmet, så det blev lidt mindre undersættelse. I sig selv er programmet enkelt:

1. Tag imod glosers og den tekst, der skal oversættes.
2. Del tekst op i enkelte ord.
3. Mere tekst?
4. Videre i tekst!
5. Oversæt et ord ad gangen.
6. Videre i data!
7. Skriv oversættelse.
- 8.-18. Data.

```

1 DIMA$(66),B$(66):FORA=1TO66:READA$(A):NEXT:
  FORA=1TO66:READB$(A):NEXT:PRINT"f5f9TAXI"
2 A=INT(RND(1)*66)+1:B=INT(RND(1)*66)+1
3 PRINT"f7DU ER I OMEGNEN AF":PRINT"f6f9"A
  $(A):PRINT"f7DIN KUNDE SKAL TIL":PRINT"f3f9
  "A$(B)
4 PRINT"f1f9TID:"C"v."D
5 IFA=BTHEN14
6 INPUT"f7RETNING(N/Ø/S/V)":A$
7 IFA$="N"THENE=VAL(LEFT$(B$(A),2))
8 IFA$="Ø"THENE=VAL(MID$(B$(A),3,2))

```

```

9  IFA$="S"THENE=VAL(MID$(B$(A),5,2))
10 IFA$="V"THENE=VAL(RIGHT$(B$(A),2))
11 IFE=0THEN13
12 A=E:D=D+10:IFD=60THENC=C+1:D=0
13 PRINT"CCCCCCCCCCCCCCCCCCCCCCCC":GOTO3
14 F=F+1:IFF < 3THENPRINT"f5f9NY KUNDE":GO
    TO2
15 A=500-C*60-D:IFA < 0THENA=0
16 PRINTA"POINTS"
17 DATAMØRDRUP,SNEKKERSTEN,HUMLEBAEK,ES
    PERGAERDE, NIVERØD, ULLERØD, LILLERØD, S
    LUTTERUP
18 DATAHØRSHOLM,LYNGE,ALLERØD,BIRKERØD,
    HØSTERKØB,VEDBAEK,BURESØ,LYNGE,FARUM,
    BIRKERØD
19 DATASØLLERØD,JAEGERSBORG,FARUM,HOLTE
    ,BREDE,EREMITAGEN,KNARDRUP,VAERLØSE,H
    ARESKOV
20 DATASORGENFRI,ORDRUP,SMØRUMNEDRE,BA
    LLERUP,HJORTESPRING,GLADSAXE,GENTOFTE,
    BALLERUP
21 DATAEJBY,HUSUM,NØRREBRO,VRIDSLØSEMAG
    LE,HERSTEDVESTER,HVISSINGE,VANLØSE,FRED
    ERIKSBERG
22 DATAKLØVERMARKEN,MARBJERG,HEDEHUSE
    NE,TAASTRUP,ALBERTSLUND,BRØNDBY,HVIDO
    VRE,SYDHAVNEN
23 DATASUNDBYØSTER,ISHØJ,ISHØJ,BRØNDBY,AV
    EDØRE,TAARNBY,TØMMERUP,KARLSLUNDE,M
    OSEDE
24 DATAHUNDIGE,KONGELUNDEN,MAGLEBY,KAR
    LSTRUP,MOSEDE,JERSIE
25 DATA00020300,00000401,01040500,02000003,0300
    0600,05000900,00081100,00091207,06001308
26 DATA00111600,07121710,08131811,09141912,0000

```



```

2013,00160000,10170015,11182116,12192217
27 DATA13202318,14002419,17222600,18232721,1924
2822,20002923,00263000,21273125,22283226
28 DATA23293327,24003428,25310000,26323530,2733
3631,28343732,29003833,31364000,32374135
29 DATA33384236,34004337,00404700,35414839,3642
4940,37435041,38445142,00005243,00460000
30 DATA00470045,39485346,40495447,41505548,4251
5649,43525750,44005851,47546000,48556153
31 DATA49560054,50570055,51586256,52006357,0060
6400,53616559,54000060,57630000,58000062
32 DATA59656600,60000064,64000000

```

NB! Prøv også

```

1 PRINT"sf3f9OPLØSNING
  ImmmmmmmmmmmPRIMFAKTORER":PRINT:IN-
  PUT"TAL";A:PRINT:PRINTA"=";:B=2
2 IFA/B=INT(A/B)THENA=A/B:PRINTB"*";:IFA>1
  THEN2
3 IFA>1THENB=B+1:GOTO2
4 PRINT"vm":RUN

```

# LEKTION 17

## DEFINERBARE FUNKTIONER

I lektion 15 lavede vi et program, der alfabetiserede ud fra de enkelte karakterers talværdier, ASCII-koden, som du her i bogen kan finde opremset i appendikset. Vi kan også få maskinen til selv at opgive dem. Ordren

```
?ASC("A")
```

betyder således: Opgiv ASCII-koden for A, som er 65. Omvendt betyder

```
?CHR$(65)
```

skriv den karakter, hvis ASCII er 65, altså et A. Funktionen er praktisk i forbindelse med

```
GET
```

Tag programmet fra sidste lektion, hvor vi skulle taste N, Ø, V eller S *fulgt af RETURN*. Her kunne vi i stedet for INPUT have haft GET. Prøv

```
1 INPUT"RETNING(N/Ø/S/V)";A$:PRINTA$
```

og derefter

```
1 PRINT"RETNING(N/Ø/S/V):?"  
2 GETA$:IFA$=""THEN2  
3 PRINTA$
```

Nu kan vi meget praktisk undvære RETURN, hvad der blandt andet gør programmet lettere tilgængeligt for den »ikke-indviede« bruger. Læg mærke til, at GET ikke »venter« som INPUT. Derfor har vi spærret den inde i en uendelig løkke, der kun brydes, når noget indtastes (når A\$ ikke længere er den tomme streng). Vi kan nu have

```
1 GETA$:IFA$=" "THEN1
2 IFA$=CHR$(133)THEN A$=CHR$(144)
3 IFA$=CHR$(137)THEN A$=CHR$(5)
4 IFA$=CHR$(134)THEN A$=CHR$(28)
5 IFA$=CHR$(138)THEN A$=CHR$(159)
6 IFA$=CHR$(135)THEN A$=CHR$(156)
7 IFA$=CHR$(139)THEN A$=CHR$(30)
8 IFA$=CHR$(136)THEN A$=CHR$(31)
9 IFA$=CHR$(140)THEN A$=CHR$(158)
10 PRINT A$;:GOTO1
```

ASCII 133-140 er de 8 FUNKTIONS-TASTER i højre side af tastaturet mærket f1-8. Lige numre får du med SHIFT. De er programmerbare, og du har netop programmeret dem! Vi kunne også have haft

```
1 GETA$:IFA$=" "THEN1
2 IFASC(A$)<133ORASC(A$)>140THEN12
3 ONASC(A$)-132GOTO4,6,8,10,5,7,9,11
4 A$=CHR$(144):GOTO12
5 A$=CHR$(5):GOTO12
6 A$=CHR$(28):GOTO12
7 A$=CHR$(159):GOTO12
8 A$=CHR$(156):GOTO12
9 A$=CHR$(30):GOTO12
10 A$=CHR$(31):GOTO12
11 A$=CHR$(158):GOTO12
12 PRINT A$;:GOTO1
```

## Udtrykket

ON A GOTO B,C,D,... E

betyder, at værdien af variablen A bestemmer, om der skal springes til linje B,C,D osv. Hvis A er 1, vælges den *første* mulighed (det første DESTINATIONS-TAL), hvis 2 den anden, og så fremdeles. De to versioner er selvfølgelig typiske demonstrations-programmer, anskuelige snarere end fikse. Pointen er imidlertid, at maskinen stiller så mange forskellige udtryksmåder til rådighed for brugeren, at det skulle være muligt i hvert enkelt tilfælde at finde den helt rigtige formulering. En anden fiks ting er

DEFFN

Bogstaverne står for »definer funktion«, og ordren sætter dig faktisk i stand til i begyndelsen af et program at definere dine egne funktioner. F.eks. kan du jo få brug for en, der vælger et tilfældigt tal mellem 1 og et opgivet tal. Du kunne så have

DEFFNA(X)=INT(RND(1)\*X)+1

Maskinen protesterer

?ILLEGAL DIRECT  
ERROR



= kan ikke bruges uden for programmer, så vi stopper den ind i et:

```
1 DEFFNA(X)=INT(RND(1)*X)+1
2 INPUTB
3 PRINTFNA(B)
```

Her er

A navnet på funktionen (du kunne jo have flere definerede i samme program).

X en PSEUDOVARIABEL, der viser, hvad der sker med det tal, du senere kommer i parentes.

B tallet, det i selve programmet går ud over.

Du kunne også have haft en »fast terning« eller måske to:

```
1 DEFFNA(X)=INT(RND(1)*6)+1:DEFFNB(X)=INT(RND
  (1)*6)+1+INT(RND(1)*6)+1
2 PRINTFNA(X)
3 PRINTFNB(X)
```

Funktion A slår med en terning, B med to. Læg mærke til, at argumentet nu ingenting betyder, men kun er med af hensyn til syntaksen.

**NB! Prøv**

```
1 A$=A$+CHR$(INT(RND(1)*10)+48):A=A+1:PRIN
  TA$:FORB=1TO500+A*100:NEXT
2 INPUT"hTAL";B$:IFB$=A$THEN1
3 PRINT"NEJ, "A$:PRINTA*100"POINTS"
```

## LEKTION 18

### ANDRE FUNKTIONER

I første del af denne bog gennemgik vi begyndelsesgrundene i programmeringssproget BASIC. Vi opdagede, hvad en variabel og et program var, hvordan tastaturet fungerede, samt enkle funktioner og ordrer som GOTO, INPUT, IF, relationer og logiske operationer. I anden del er vi nået et stykke videre, idet vi har stiftet bekendtskab med FOR-NEXT-løkker, strengdeling, subrutiner og sæt. Vore programmeksampler har spændt fra simple beregningsprogrammer over tal- og tekstspil til en simulation, en database, alfabisering og tekstbehandling.

Derimod er vi gået uden om de bekendte TV- og automatspil, VICs lyd- og musikkapacitet, eller kort og godt: Det meste af det, der fik os til at vælge en VIC fremfor en ZX81: Hurtighed, farver og lyd. Når vi har forsømt disse faciliteter så langt, skyldes det, at vi her ikke kan klare os med »almindelig« BASIC, vi må »om bag kulisserne« og erfare lidt om, hvordan maskinen virker, og således skaffe os en kontrol med ikke mindst skærmen, som det blotte og bare kendskab til tastaturet og BASICs syntaks ikke kan give os. Herom BOGENS TREDJE DEL.

I resten af denne del skal vi samle nogle løse ender op. I lektion 19 skal vi lære at oplagre vore programmer på bånd. I lektion 18 skal vi se på nogle forsømte funktioner. Mange af disse er selvfølgelig hovedsagelig til glæde for matematikeren, og dem skal vi gå forholdsvis let hen over. F.eks. er der de såkaldt trigonometriske funktioner:

SIN(X) – SINUS

COS(X) – COSINUS

TAN(X) – TANGENS  
ATN(X) – BUETANGENS

LOG(X)

og

EXP(X)

bruges til beregning med naturlige logaritmer. Prøv

```
?12*34
```

og derefter

```
?EXP(LOG(12)+LOG(34))
```

Noget mere brugbar for almindelige dødelige er

ABS(X)

Prøv

```
1 A=INT(RND(1)*100):B=INT(RND(1)*100):PRINTA"*"B  
  ;:INPUTC:IFC=A*BTHEN1  
2 PRINT"DU RAMTE"ABS(C-A*B)"FORKERT":GO  
  TO1
```

Her er det ikke meningen, at der skal svares ubetinget rigtigt, men at man gør et skøn og derefter ser, hvor nær man kom. Havde vi nu simpelt hen skrevet  $C-A*B$ , ville vi have fået et negativt tal halvdelen af gangene. ABS smider fortegnet væk.

POS(X)

opgiver, hvor markøren er henne i linjen. Prøv

?TAB(10)POS(0)

Også her er funktionens argument uden betydning.

Vi har allerede været igennem de fleste fejlmeddelelser. Andre, du kan risikere, er

ILLEGAL QUANTITY = for stort argument for den pågældende funktion, altså f.eks. CHR\$(300).

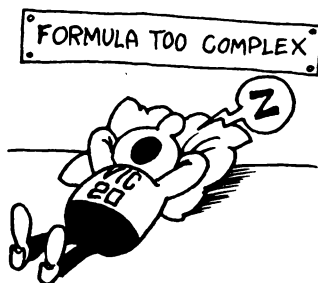
UNDEF'D FUNCTION = du har glemt en DEFFN!

Herudover har maskinen visse begrænsninger. Nogle er selvfølgelig. Man *kan* faktisk ikke dividere med 0, og et forsøg herpå giver

DIVISION BY ZERO

Andre er begrænsninger ved selve maskinen. Det kan (sjældent) ske, at et udtryk bliver for kompliceret for maskinen at arbejde med. Så siger den

FORMULA TOO COMPLEX





Del udtrykket op i to mindre, og det går godt igen. Maskinen kan heller ikke arbejde med strenge på mere end 255 karakterer. Her lyder fejlmeddelelsen

STRING TOO LONG

og vi må ty til sæt.

Måske har du også prøvet at rette i en programlinje og så komme hurtigt ned i bunden med RETURN. Så er du garanteret også af og til endt med en OUT OF DATA. Hvorfor? Jo, da markøren nåede READY, og du derefter tastede RETURN, opfattede maskinen det som den direkte ordre READ Y, og det kan maskinen selvfølgelig ikke, eftersom der ikke er nogen DATA-linjer at læse Y fra! Prøv med SHIFT RETURN i stedet!

Endelig er der faktisk to hele ordrer, vi slet ikke har brugt! Den ene er

REM

der står for REMark, bemærkning, og den betyder faktisk, at maskinen *ikke* skal gøre det, der står i ordren, altså: Dette er ingen ordre til maskinen, men en note til oplysning for den, der læser programmet.

Så har vi

LET

der helt kan undværes.

LETA=27

betyder nemlig ganske det samme som

A=27

## NB! Prøv

```
1  REM LØVEFAMILIER AF FORSKELLIG
   STØRRELSE ("O" < "W" < "Q") OG GAZELLE
   FLOKKE (*)
2  DIMA%(23,24),B%(23,24):FORA=1TO100 : B%
   (INT(RND(1) * 21)+2,INT(RND(1)* 22)+2)=1:NEXT
3  FORA=1TO100:B%(INT(RND(1) * 21)+2,INT (RND(1)
   * 22)+2)= 9:NEXT:PRINT"h"
4  B=B+1:PRINT"hf4f9ANNO"B:A= 0 : C = 0 :FORD
   =2TO22:FORE=2TO23:A% (D,E) = B% (D,E)
5  IFA%(D,E)=0THENPRINT"m";
6  IFA%(D,E)=1THENPRINT"f3O";:A=A+1
7  IFA%(D,E)=2THENPRINT"f3W";:A=A+1
8  IFA%(D,E) > 2AND A% (D,E) < 9THENPRINT"f3Q";
   :A=A+1
9  IFA%(D,E)=9THENPRINT"f1*";:C=C+1
10 NEXT:NEXT:PRINT"f4f9LØVE"A"vmGAZELLE"C"
    vm";
11 FORD=2TO22:FORE=2TO23:IFA% (D,E)=0ORA%
   (D,E)= 9THEN18
12 IFA%(D-1,E)=9THENB% (D-1,E)= 0 : B% (D,E)=A%
   (D,E)+1
13 IFA%(D,E-1)= 9THENB% (D,E-1)= 0 : B% (D,E)= B%
   (D,E)+1
14 IFA%(D,E+1)= 9THENB% (D,E+1)= 0 : B% (D,E)= B%
   (D,E)+1
15 IFA%(D+1,E)= 9THENB% (D+1,E)= 0 : B% (D,E)= B%
   (D,E)+1
16 IFA%(D,E) > 3THENB% (D,E)=A% (D,E)-2: B%
   (INT(RND(1) * 21)+2,INT(RND(1)* 22)+2)=2
17 B%(D,E)= B% (D,E)-1
18 NEXT:NEXT:FORD=1TOC :E=INT(RND(1) * 21)
   +2:F=INT(RND(1) * 22)+2
19 IFB%(E,F)=0THENB%(E,F)=9
20 NEXT:GOTO4
```

## LEKTION 19

### SAVE

VIC er en farve-datamat til tre tusind, og det er selvfølgelig en af dens væsentlige fordele. Samtidig eksisterer der imidlertid en masse tilbehør, hvoraf en del er næsten lige så dyrt, eller endog dyrere end selve maskinen. Det er derfor et nærliggende spørgsmål: Hvor meget behøver jeg?

Et svar er naturligvis: Ikke noget. Alle de i bogen gennemgængede funktioner og programmer kræver kun grundmodellen. To ting er dog mildest talt gode at have: Kassettebåndstationen, så vi ikke behøver at taste alle vore programmer ind på ny, hver gang vi tænder for maskinen, og en hukommelsesudvidelse. Og så kan det faktisk kun betale sig at tage en 16K med det samme. Vi står så ganske vist med en 5000-kroners datamat, men til gengæld også en, der opfylder alle vore rimelige behov. Selvfølgelig er en printer både sjov og praktisk, den er jo et helt lille trykkeri for sig. Men nødvendig er den ikke. Og 5000 ekstra er mange penge. Vi skal her forudsætte, at du har kassettestationen, og lære at gemme vore programmer på bånd. Har du ingen, kan du indtil videre bare springe denne lektion over.



Indtast nu f.eks. TAXI-spillet fra lektion 16. Sæt bånd i båndoptageren. Vær sikker på, at det ikke er sikret mod sletning ved, at den lille tap øverst til venstre på båndsiden er trykket ud. Er det sket, kan du gøre det godt igen med et stykke tape. Check ligeledes, at båndet er spolet tilbage. Nu taster du

SAVE”TAXI”

eller blot

SAVE

fulgt af RETURN. I det første tilfælde har du givet programmet et navn, fordi du har i sinde at have flere programmer på et bånd. Lad os blot vælge denne mulighed. Maskinen skriver

PRESS RECORD & PLAY  
TAPE

Du holder REC nede på båndoptageren og trykker PLAY ned, til den låser, og båndet kører. Maskinen siger

OK

SAVING TAXI

og efter et stykke tid

READY.

Læg mærke til, at båndet er standset. Tryk nu på STOP og derefter REW. STOP igen, når båndet er spolet tilbage. Du

har nu formodentlig en optagelse, men vil du være sikker, kan du VERIFICERE det. Tast

VERIFY

fulgt af RETURN. Maskinen forlanger

PRESS PLAY ON TAPE

og kvitterer med OK, når du trykker PLAY, endvidere

SEARCHING

dvs. »jeg søger«, nemlig efter et program at verificere. Da vi ikke har givet besked om, hvilket program den skal lede efter, tager den blot det første på båndet. Efter en tid får vi

FOUND TAXI

så

VERIFYING

og endelig

OK

READY.

Det eneste, der kan gå galt, er sådan set en fejl i båndet, der giver VERIFY ERROR. Så må vi bare prøve igen med et andet bånd. Spol tilbage og tryk på STOP, og så en gang til, denne gang for EJECT. Låget springer op, og vi kan tage vores bånd, anbringe det et sted uden for fjernsynets umiddelbare nærhed (et TV er en elektromagnetisk dingensnot og kan

slette båndet) og slukke for datamaten. Nu leger vi, at det er blevet torsdag, og tænder igen. Vores program er naturligvis pist væk. Læg nu båndet i båndoptageren og tast

LOAD

Der er jo kun det ene program på båndet. Igen får vi at vide, at vi skal trykke på PLAY, vi gør det, maskinen svarer OK og går i gang med at lede. Vi får

FOUND TAXI

og derefter

LOADING

samt et READY, når den er færdig. Tast nu LIST og RUN, og vi har vores program som før. Havde vi nu ikke verificeret, og båndet havde været defekt, havde vi her fået LOAD ERROR. Så havde det bare været for sent . . .

NB! Prøv

```
1 A$="1V2":PRINT"f9TIPSKUPONs":FORA=1TO13: B
  =INT(RND(1)*3)+1:PRINTTAB(B) MID$(A$,B,1)
  :NEXT
```

# LEKTION 20

## SPIEL

```
1 PRINT"hDU FØRER RUMSKIBETmmmmCO
  MMODORE":A=1000
2 B=1000+RND(1)*1000:C=0:D=0:PRINT"FJENDTLI
  GT RUMSKIB ImmSIGTE"
3 PRINT"f6f9POINTS"E:PRINT"sf7AFSTAND"B:PRIN
  T"RELATIV HASTIGHED"F:PRINT"ENERGI"A
4 PRINT"KRAFTSKJOLD"G:PRINT"SKADE I SKIB"I
  NT(H):PRINT"SKADE I ANDET SKIB"INT(C)
5 IFRND(1) < .9THENPRINT"sf5f9FJENDEN SKYDE
  R":H=H+RND(1)*(1000-B-G)/10
6 PRINT"sf3F1 HASTIGHED":PRINT"F3 KRAFTSKJ
  OLD":PRINT"F5 LASER":PRINT"F7 FASER"
7 PRINT"sf9ORDRE?"
8 GETA$:IFA$=""THEN8
9 I=ASC(A$)-132:IFI=1THENINPUT"HASTIGHED
  (0-100)";F
10 IFI=2THENINPUT"KRAFTSKJOLD (0-100)";G
11 IFI=3THENA=A-10:C=C+RND(1)*(1000-B)/10
12 IFI=4THENA=A-20:C=C+RND(1)*(1000-B)/5
13 A=A-(F+G)/10:B=B-F-RND(1)*100:IFC < 0THENC
  =0
14 IFH < JTHENH=J
15 IFC >=100THENPRINT"hf9FJENDEN ØDELAGT":
  E=E+(25-D)*100:J=H:PRINT"NYT":GOTO2
16 IFA <=0THENPRINT"f9ENERGI 0":GOTO20
17 IFB <=0THENPRINT"f9SAMMENSTØD":GOTO20
18 IFH >=100THENPRINT"f9COMMODORE ØDELA
  GT":GOTO20
```

```

19 D=D+1:PRINT"h":GOTO3
20 PRINT"f9POINTS"E

```

Ovenstående program indeholder en stor del af de ordrer og funktioner, vi har gennemgået i denne bogs to første dele. Gennemgå og forstå det. Prøv at lave det om. Som en hjælp er her en liste over variabler:

```

A ENERGI
B AFSTAND
C SKADE I ANDET SKIB
D OMGANG
E POINTS
F RELATIV HASTIGHED
G KRAFTSKJOLD
H SKADE I SKIB
I ORDRE
J SIDSTE SKADE I SKIB

```

Prøv på samme måde

```

1 FORA=1TO6:A(A)=INT(RND(1)*10)+1:NEXT:PRINT
  "FIND PRINSESSEN, INDENUHYRET FINDER DI
  G"
2 PRINT"f7RETNING(N/S/V/Ø)?"
3 GETA$:IFA$=""THEN3
4 IFA$="N"THENA(1)=A(1)-1
5 IFA$="S"THENA(1)=A(1)+1
6 IFA$="V"THENA(2)=A(2)-1
7 IFA$="Ø"THENA(2)=A(2)+1
8 IFA(1)=A(3)ANDA(2)=A(4)THENPRINT"f5f9SPIST!":E
  ND
9 IFA(1)=A(5)ANDA(2)=A(6)THENPRINT"f5f9ELSKED
  E!":PRINT"POINTS"(50-E)*10:END
10 IFRND(1)<.75THEN15

```



```

11 PRINT"f3UHYRET ER I ";:IFA(3) < A(1)THEN A
    (3)=A(3)+1:PRINT"NORD"
12 IFA(3) > A(1)THEN A(3)=A(3)-1:PRINT"SYD"
13 IFA(4) < A(2)THEN A(4)=A(4)+1:PRINT"VEST"
14 IFA(4) > A(2)THEN A(4)=A(4)-1:PRINT"ØST"
15 IFRND(1) < .75 THEN 20
16 PRINT"f6PRINSESEN ER I ";:IFA(5) < A(1)THEN A
    (5)=A(5)-1:PRINT"NORD"
17 IFA(5) > A(1)THEN A(5)=A(5)+1:PRINT"SYD"
18 IFA(6) < A(2)THEN A(6)=A(6)-1:PRINT"VEST"
19 IFA(6) > A(2)THEN A(6)=A(6)+1:PRINT"ØST"
20 FOR A=1 TO 6:IFA(A)=0 THEN A(A)=10
21 IFA(A)=11 THEN A(A)=1
22 NEXT A:A=ABS(A(1)-A(3)):B=ABS(A(2)-A(4)):C=ABS
    (A(1)-A(5)):D=ABS(A(2)-A(6))
23 IFA < 2 AND B < 2 THEN PRINT"f3f9MØRK SKYG
    GE"
24 IFA < 4 AND B < 4 THEN PRINT"f3f9TUNGE SKRID
    T"
25 IFC < 2 AND D < 2 THEN PRINT"f6f9SKRIG"
26 IFC < 4 AND D < 4 THEN PRINT"f6f9LØBENDE FØD
    DER"
27 PRINT"f8CCCCCCCCCCCCCCCCCCCC":E=E+1:
    GOTO 2

```

Så enkle disse spil end er, fortæller de dog noget om »mekanikken« i et program. Prøv programmet og besvar følgende spørgsmål:

1. Hvordan ved maskinen, hvem der er hvor i »labyrinten«?
2. Hvorfor går uhyret og prinsessen langsommere end du?
3. Hvordan kan maskinen fortælle dig, om f.eks. uhyret er nord eller syd for dig?
4. Det hele foregår egentlig på overfladen af en kugle. Hvordan det?

5. Hvordan kan maskinen fortælle dig, at f.eks. prinsessen er nær ved dig?
6. Uhyret jager dig, prinsessen løber væk fra dig. Hvordan ordner maskinen det?
7. Somme tider er prinsessen skiftevis f.eks. nord og syd for dig. Hvorfor? Hvordan kan du så finde hende?

Find frem, hvad du synes, er svagheder ved dette spil, og lav det om, til det passer dig. Lav andre spil. Hvis du nøjes med at læse de første 20 lektioner igennem, lærer du intet. *Alle principperne skal bruges af dig i dine egne programmer!!* Det er den gyldne regel for hele denne bog, og hvis du ikke overholder den, kan du lige så godt lægge bogen fra dig med det samme. Du lærer alligevel ikke at svømme uden at blive våd.

Hvis du derimod nøje har læst de første 20 lektioners gennemgang og beskrivelse af funktionerne og brugt hver eneste af dem i det mindste et par gange, skulle du faktisk allerede nu være i besiddelse af pæne forudsætninger i BASIC. Du har dog endnu langt fra udnyttet din maskines muligheder fuldt ud. At du nu kender til de store tandhjul, betyder ikke, at du har fuld kontrol over maskinen. Der er trods alt langt fra tal på en skærm til et fungerende TV-spil. Denne vej vil vi imidlertid tilbagelægge i tredje del af bogen.

Så se lige engang tilbage; har du nu husket alt (eller i al fald det meste)? Godt, så fanger vi an.

**NB! Prøv**

```

1 FORA=1TO6:READA$(A):A(A)=100:NEXT:A=1000
2 PRINT"hf6f9ANTALmf3AKTIEmf7KURS":PRINT
  :FORB=1TO6:PRINT"f6f9" B(B) "f3" B ;A$(B)
  "f7" A(B):NEXT
3 PRINT:PRINT"KAPITAL:"A:PRINT:INPUT
  "KØB/SALG (K/S) ";A$:IFA$="K"THEN6
4 IFA$="S"THEN8
5 GOTO10

```

```

6 GOSUB14:IFA(B)*C > A THEN6
7 B(B)=B(B)+C:A=A-A(B)*C:GOTO10
8 GOSUB14:IFC > B(B) THEN8
9 B(B)=B(B)-C:A=A+A(B)*C
10 FORB=1TO6:A(B)=A(B)+INT(RND(1)*21)-10:IFA(B)
    < 10 THENA(B)=10
11 NEXT:A=A-10:IFA < 0 THENPRINT:PRINT"DU ER
    FALLIT":END
12 IFA > 10000 THENPRINT:PRINT"DU ER BØRSMA
    TADOR":END
13 GOTO2
14 PRINT:INPUT"AKTIENUMMER";A$:IFASC(A$)
    < 49ORASC(A$) > 54 THEN14
15 B=VAL(A$):PRINT:INPUT"HVOR MANGE";C:RE
    TURN
16 DATAJENSEN,OLSEN,KLAUSEN,FEDESEN,TOM
    SEN,BORGEN

```



*TREDJE DEL*  
*Billede og lyd*



# LEKTION 21

## BITS OG BYTES

```
1 A$="nsøv"
2 PRINTMID$(A$,INT(RND(1)*4)+1,1)"f1X";:FORA=1T
  O100:NEXT:PRINT"vmv";:GOTO2
  100:NEXT:PRINT"vmv";:GOTO2
```

Som du vil se, er der kommet en flue ind i maskinen. Dette undgås bedst ved altid at holde vinduerne lukkede, mens man arbejder med den.

En vis primitiv illusion af bevægelse er altså mulig uden andre forkundskaber end dem, vi er i besiddelse af på nuværende tidspunkt. Skal vi videre, må vi imidlertid vide lidt mere om, hvordan maskinen *fungerer*.

I skolen lærte vi at regne med tal på mere end et ciffer ved f.eks. at opsplitte 1234 i

1 tusind  
2 hundrede  
3 tiere  
4 enere

Vores »almindelige« talsystem er et titalssystem, dvs. det består af 10 tal: 0, 1, 2, 3, 4, 5, 6, 7, 8 og 9. Når vi har talt op til 9, får vi 0 igen på ener-pladsen og begynder at tælle på tier-pladsen med 1, op til 9 tiere og 9 enere, hvor vi må tage hundrede-pladsen i brug.

Vi kan imidlertid også lave et talsystem med kun to tal, 0 og 1. Nu skal vi skifte plads allerede, når der står 1 på denne plads, det højest tilladelige. Vi skifter altså første gang plads

ved 2, der bliver 10. Derefter bliver 3 til 11 og 4 bliver til 100. Vi har altså

| TOTALSSYSTEM | TITALSSYSTEM |
|--------------|--------------|
| 0            | 0            |
| 1            | 1            |
| 10           | 2            |
| 11           | 3            |
| 100          | 4            |
| 101          | 5            |
| 110          | 6            |
| 111          | 7            |
| 1000         | 8            |
| 1001         | 9            |
| 1010         | 10           |
| 1011         | 11           |
| 1100         | 12           |
| 1101         | 13           |
| 1110         | 14           |
| 1111         | 15           |
| 10000        | 16           |

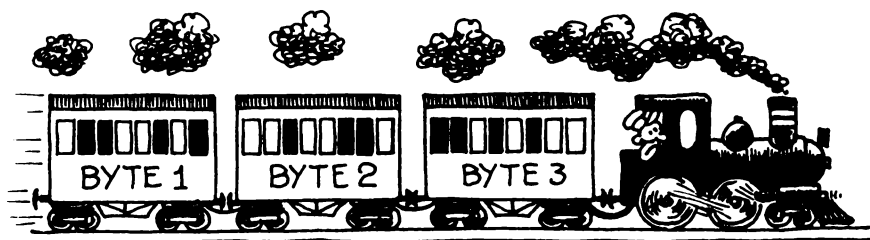
Upraktisk? Ikke for en maskine. Nu kan nemlig ethvert tal udtrykkes ved en serie elementer, der enten er tændte eller slukkede. Tallet 13 kræver fire lamper, og 1101 udtrykkes

#### TÆNDT TÆNDT SLUKKET TÆNDT

altså tændt for 1 og slukket for 0. Totalssystemet hedder på engelsk BINARY SYSTEM, og derfor kalder man et element, der ved at være tændt eller slukket udtrykker et ciffer i totalssystemet, BINARY DIGIT (digit = ciffer) eller simpelt hen BIT. For at få »rigtige« tal, er disse bits ordnet i enheder på 8 bits, og disse enheder kaldes en BYTE. En byte kan



(regn efter) udtrykke et tal mellem 0 og 255. Nu ved du, hvad det vil sige, at du, når du tænder for maskinen, har 3583 bytes til rådighed, og hvorfor en byte cirka kunne oversættes med en karakter. Der er 256 ASCII-koder, hvilket vil sige, at en hvilken som helst af ASCII-systemets karakterer kan udtrykkes ved et tal mellem 0 og 255 og altså af 1 byte. Læg mærke til, at det ikke er ligegyldigt, hvilke bits der er tændt i en byte. 10110010 er noget andet end 11001001. På samme måde skal flere bytes naturligvis også arbejde sammen, når hele BASIC-sprogets utallige muligheder skal udtrykkes. Hver byte har sin plads eller ADRESSE. MASKINE er jo heller ikke det samme som ANEMISK.



Bytes tælles i K. 1 K eller kilo-byte er 1024 bytes. En 16K hukommelsesudvidelse vil altså give dig godt og vel 16000 bytes ekstra at arbejde med. Alt dette er RAM eller Random Access Memory, dvs. hukommelse, som du kan bruge, som du vil, til programmer, variabler, etc. Men maskinen indeholder naturligvis også en for-programmeret viden om, hvordan man lægger sammen, uddrager kvadratrødder, eller blot oversætter dit halv-engelske BASIC til »binærkode« eller »maskinkode«. Denne form for hukommelse kaldes ROM eller Read Only Memory. VIC har 20K ROM, så det er ikke så lidt, den ved, endnu før du har fortalt den noget.

RAM-adresserne er opdelt i større »byer«, hver med deres egen funktion. Der er særlige områder for oplagring af

programmer, variabler, skærbilledet, etc. Herudover er der de såkaldte SYSTEMVARIABLER, der fortæller maskinen noget om, i hvilken »tilstand« den er. Den »egentlige« maskine er nemlig den såkaldte CPU, »Central Processing Unit«. Det er »regneværket« i VIC, og det står i forbindelse med alle de forskellige former for hukommelse.

SYSTEMVARIABLERNE skal vi se på i næste lektion. Her skal vi skrive et lille nemt program, der kan »oversætte« et tal i totalssystemet til et tal i titalssystemet inden for intervallet 0-255, noget, vi snart skal få brug for:

```
1 INPUT A$:FOR A=1 TO 8:B=B+VAL(MID$(A$,A,1))*2p(8-A):NEXT:PRINT B
```

Husk, når du afprøver programmet, at sætte nuller forrest, sådan at der i alt er 8 cifre i inputtallet.

Vi kan naturligvis også gå den anden vej:

```
1 INPUT A:FOR B=7 TO 0 STEP -1:IF A >=2pB THEN PRINT"1";A=A-2pB:GOTO 3
2 PRINT"0";
3 NEXT
```

Prøv nogle tal igennem på de to programmer. Prøv at finde nogle regneregler for totalssystemet. Du skal føle dig forholdsvis sikker i disse tal, før du går videre . . .

**NB! Prøv**

```
1 REM FIDUS: OPTEGN OASERNES PLACERING I
  HVERT SPIL OG RID EVENTUEL TILBAGE EF-
  TER VAND
2 FOR A=1 TO 10:READ A$(A):A(A)=INT(RND(1)
  *1000):NEXT A: A=1: B=10: C=10: D=10: E=1
3 PRINT"hf4"A". DAG":PRINT:PRINT"f3DU HAR
```

```

REJST"F"KM":PRINT"DU HAR"B"L VAND"
4 PRINT"f6":IFC > 0THENPRINT"DU ER TIL
  HEST":PRINT"HESTEN ERm"A$(C)
5 IFC=0THENPRINT"DU ER TIL FODS"
6 PRINT:PRINT"f7DU ERm"A$(D):IFA > 10ANDG
  < 100THENPRINT"f9RØVERE"G"vmKM BORTE"
7 FORH=1TO10:IFABS(A(H)-F) < 10ANDC >
  0THENPRINT"f9HESTEN LUGTER VAND"
8 IFABS(A(H)-F) < 5THENPRINT"f9OASE":B=10:D
  =10:IFC > 0THENC=10
9 NEXT:IFRND(1) < .01THENPRINT"f9SANDSTO
  RM":B=0:C=C-5:D=D-5
10 PRINT:PRINT"f5VIL DU"
11 PRINT"1. REJSE HURTIGERE?":PRINT"2. REJSE
  LANGSOMMERE?":PRINT"3. HVILE MEGET?"
12 PRINT"4. DRIKKE?":PRINT"5. VENDE OM?"
  :PRINT"6. REJSE SOM FØR?"
13 GETA$:IFA$=""THEN13
14 IFASC(A$) < 49ORASC(A$) > 54THEN13
15 ONVAL(A$)GOTO16,19,23,25,30,32
16 H=H+INT(RND(1)*10):IFC > 0THENC=C-3
17 IFC=0THEND=D-2
18 D=D-1:GOTO34
19 H=H-INT(RND(1)*10):IFH < 1THENH=1
20 IFC > 0THENC=C-2
21 IFC=0THEND=D-1
22 D=D-1:GOTO34
23 H=0:IFC > 0THENC=C-1
24 D=D-1:GOTO34
25 IFB > 0THEN28
26 IFC > 0THENC=C-1
27 D=D-1:GOTO34
28 B=B-1:IFC > 0THENC=C-1
29 D=D+1:GOTO34
30 E=E*-1:IFC > 0THENC=C-1

```

```

31 D=D-1:GOTO34
32 IFC > 0THENC=C-1
33 D=D-1
34 A=A+1:F=F+(C+D+H)*E:IFF < 0THENPRINT"f
    9DU ER HJEMME IGEN":END
35 IFA < 11THEN39
36 IFF > ITHENI=I+INT(RND(1)*10)
37 IFF < ITHENI=I-INT(RND(1)*10)
38 G=ABS(F-I):IFG < 5THENPRINT"f9RØVERNE
    INDHENTEDE DIG":END
39 IFC > 10THENC=10
40 IFC < 0THENC=0
41 IFD > 10THEND=10
42 IFD <=0THENPRINT"f9DU ER DØD":END
43 GOTO3
44 DATADØENDE,MEGET SYG,SYG,UTILPAS,ME
    GET VARM, TØRSTIG, VARM, VELTILPAS,RASK,I
    TOPFORM

```

## LEKTION 22

### SYSTEMVARIABLER

Skift farve til rød. Maskinen vil skrive rødt, indtil du forlanger en anden farve. Men hvordan kan den nu vide det? Du har jo for længst sluppet farvetasten. Et eller andet sted i maskinen må den være »slået til« rød farve.

Faktisk skynder maskinen sig ned i ADRESSE 646 og ser efter, hvad der står der, hver gang den skal til at skrive en karakter. Hver farve har nemlig sin FARVEKODE:

- 0 SORT
- 1 HVID
- 2 RØD
- 3 LYSEBLÅ
- 4 VIOLET
- 5 GRØN
- 6 BLÅ
- 7 GUL

eller 1 mindre end den tast, den står på. Det får vi brug for i anden sammenhæng. Forklaringen er nu simpel: Så længe værdien i adresse 646 er 2, så længe vil maskinen skrive rødt.

Det er selvfølgelig noget af en påstand, men vi kan nu også bevise den. Vi har nemlig en funktion, der kan kigge direkte ind i en adresse, faktisk betyder

PEEK

på engelsk »kigge«. Prøv nu

PRINTPEEK(646)

2, ikke sandt? Skift farve og prøv igen. Prøv nu

POKE646,5

Maskinen skriver *READY med grønt*. Ordren POKE virker nemlig den modsatte vej: Den lægger en værdi direkte ind i en adresse. Her skiftede vi 2 ud med 5, og maskinen skriver grønt!

Prøv at slå tilbage i første dels lektion 4. Tag eksemplet

```
10 PRINT"f6VIC 20"  
20 PRINT"FRA f3COMMODORE"  
30 PRINT"f1DEN PERSONLIGE DATAMAT"
```

Vi kunne sådan set også have haft

```
10 POKE646,5:PRINT"VIC 20"  
20 PRINT"FRA ";:POKE646,2:PRINT"COMMODORE"  
30 POKE646,0:PRINT"DEN PERSONLIGE DATAMAT"
```

Ikke særlig praktisk? Nej, vel? Og her er sådan set sagens kerne. I langt de fleste tilfælde er det langt bedre at forlade sig på »almindelig« BASIC og styre maskinen fra tastaturet. Somme tider er det imidlertid ikke. Til andre kan man slet ikke opnå den virkning, man tilstræber, uden POKE.

646 er en SYSTEMVARIABLE. Vi kan sådan set ikke hente noget ud af den med POKE, som vi ikke kan få meget enklere fra tastaturet. Men tag så ADRESSE 36879. Prøv

POKE36879,25

Skærmkanten, som vi er blevet så vant til at se, forsvandt.



Det vil sige, det gjorde den sådan set ikke. Den fik bare samme farve som tekstfeltet, hvid. Prøv

POKE36879,26

Rød kant! Faktisk kan vi vælge mellem ikke mindre end 8 kantfarver og 16 skærmfarver. Vi har

#### SKÆRMFARVER

- 1 SORT
- 2 HVID
- 3 RØD
- 4 LYSEBLÅ
- 5 VIOLET
- 6 GRØN
- 7 BLÅ
- 8 GUL
- 9 ORANGE
- 10 LYS ORANGE
- 11 LYSERØD
- 12 MEGET LYSEBLÅ
- 13 LYSVIOLET
- 14 LYSEGRØN

15 OGSÅ EN SLAGS LYSEBLÅ  
16 LYSEGUL

KANTFARVER

- 1 SORT
- 2 HVID
- 3 RØD
- 4 LYSEBLÅ
- 5 VIOLET
- 6 GRØN
- 7 BLÅ
- 8 GUL

Du kan nu opnå enhver kombination ud fra formelen

$$\text{SKÆRM} * 16 + \text{KANT} - 9$$

En rød skærm med grøn kant (gisp!) ville altså kræve

$$\text{POKE}36879,3 * 16 + 6 - 9$$

eller blot

$$\text{POKE}36879,45$$

Det kan måske lyde lidt underligt, men ikke hvis vi prøver tallene igennem på vores »binær-generator« fra sidste lektion. Tag f.eks. de 45:

0 0 1 0 1 1 0 1

der kan tolkes som

0 0 1 0  
skærmfarve 2

1

1 0 1  
kantfarve 5



Igen gælder sort for 0. Hvad med 1-tallet i midten? Prøv at skifte det ud med et 0. Vi får

0 0 1 0 0 1 0 1

eller 37. Hvad skete der?

NB! Prøv også

```
1 PRINT"1":FORA=0TO255:POKE36879,A:FORB=1TO
  1000:NEXT:NEXT
```

```
1 PRINT"SKRIV ELLER LØS KODE":INPUT"(S/L)"
  ;A$:IFA$="S"THEN4
2 IFA$="L"THEN9
3 RUN
4 INPUT"KLARTEKST";A$:INPUT"KODENUMMER
  (1-25)";A
5 FORB=1TOLEN(A$):C=ASC(MID$(A$,B,1)):IFC
  =32THENPRINT"";GOTO8
6 C=C+A:IFC > 90THENC=C-26
7 PRINTCHR$(C);
8 NEXT:PRINT:RUN
9 INPUT"KODETEKST";A$:INPUT"KODENUMMER
  (1-25)";A
10 FORB=1TOLEN(A$):C=ASC(MID$(A$,B,1)):IFC
  =32THENPRINT"";GOTO13
11 C=C-A:IFC < 65THENC=C+26
12 PRINTCHR$(C);
13 NEXT:PRINT:RUN
```

## LEKTION 23

### SKÆRMEN

Prøv

```
1 PRINT"h"
2 POKE36879,8
3 POKE8000,1:GOTO3
```

De første to linjer indeholder ingen hemmeligheder for os. Skærmen slettes og farves sort. Men 3? Et hvidt A kommer til syne. Vi har ikke tastet noget A. Vi har POKet det direkte ind i den adresse, der oplagrer, hvad der står på det pågældende felt på skærmen. Prøv nu (efter RUN STOP og RUN STOP/RESTORE)

```
1 PRINT"h"
2 POKE36879,8
3 FORA=7680TO8185:POKEA,1:NEXT:GOTO3
```

Nemlig, hvad der står på skærmen, oplagres i adresserne 7680-8185 efter formlen

`LINJE*22+PLADS+7657`

Hvad er det så, der står i adresserne? Ja, ASCII er det åbenbart ikke, for ASC(1) er ikke A. Det er de såkaldte SKÆRMKODER, som du kan finde i denne bogs appendiks.

Men hvor står det, at Aet skal være hvidt? Ingen steder, og det er det sådan set heller ikke, det er farveløst. Skal vi have farve på, må vi poke den ind i en tilsvarende del af maskinen, der oplagrer hvert enkelt skærmfelts farve. Den

strækker sig over adresserne 38400-38905 og går efter formlen

LINJE\*22+PLADS+38377

Farvekoderne har vi fra lektion 22 og siger

```
1 PRINT"h"
2 POKE36879,25
3 POKE10*22+10+7657,83
4 POKE10*22+10+38377,2:GOTO4
```

eller

```
1 Slet skærm
2 Helt hvid skærm
3 Hjerte på 10. linje 10. plads
4 Karakteren på 10. linje 10. plads rød
```

Vi skulle jo imidlertid også gerne have lidt bevægelse i tingene. Prøv

```
1 PRINT"h"
2 POKE36879,25
3 FORA=1TO22
4 POKE10*22+A+7657,81
5 POKE10*22+A+38377,2
6 FORB=1TO100:NEXT
7 POKE10*22+A+7657,32
8 NEXT
9 GOTO3
```

eller

```
1 Slet skærm
```

- 2 Helt hvid skærm
- 3 FOR
- 4 Bold på 10. linje A. plads
- 5 Karakteren på 10. linje A. plads rød
- 6 Vent 1/10 sekund
- 7 Mellemrum på 10. linje A. plads
- 8 NEXT
- 9 Om igen

Når vi har set på bolden i 1/10 sekund, trykker vi et mellemrum oven på den, altså sletter den, og trykker den så igen en plads længere til højre. Illusionen er bevægelse!

Når bolden har nået højrekant, begynder den forfra i venstre side. Men kunne det nu ikke være morsommere, hvis den »stødte mod« højrekanten og løb den anden vej? Prøv

```
1 PRINT"h"
2 POKE36879,25
3 A=1
4 B=1
5 POKE10*22+A+7657,81
6 POKE10*22+A+38377,2
7 FORC=1TO100:NEXT
8 POKE10*22+A+7657,32
9 A=A+B
10 IFA=1ORA=22THENB=-B
11 GOTO5
```

Vi har her udskiftet kontrolvariablen A med en almindelig variabel, der bliver B større for hver omgang gennem løkken. I første omgang er B 1, og A gennemløber derfor værdierne 2, 3, 4 osv. Men når A bliver 22 (når højrekant), bliver B til -B, altså 1 til -1, og A antager værdierne 21, 20, 19 etc. Til sidst er A 1, på hvilket tidspunkt -B bliver -(-B), altså

1 igen, og bolden atter drager mod højre. Når du er sikker på, at du har forstået dette til bunds, kan du gå videre med

```
1 PRINT"h":POKE36879,30:A=1:B=1:C=1:D=1
2 POKEA*22+B+7657,81:POKEA*22+B+38377,2:FORE
  =1TO100:NEXT:POKEA*22+B+7657,32:A=A+C:B=B
  +D
3 IFA=1ORA=23THENC=-C
4 IFB=1ORB=22THEND=-D
5 GOTO2
```

eller

- 1 Gør skærmen parat og initialiser
- 2 Tegn og mal bold, vent, slet og flyt bold
- 3 Skift flag for A
- 4 Skift flag for B
- 5 Om igen

Læg mærke til, at vi kalder C og D for FLAG. Flag er vore egne private systemvariabler. Når flag 2 er hejst på stang 646, ved maskinen, at den skal skrive med rødt. På samme måde fortæller vi nu maskinen, at når flag D er hejst (1), skal bolden mod højre, når det er strøget (-1), mod venstre.

Vi har nu lært lidt om, hvordan vi får ting til at bevæge sig på skærmen. Eksperimentér selv videre. I næste lektion skal vi prøve, om vi ikke kan få et lille spil ud af vores dansende bold. Døren er åbnet på klem ind til TV-spillene. Lad os se, om vi ikke kan få den helt op.

NB! Prøv lige følgende

```
1 PRINT"hf3HAR DU FUNDET FREM TILEN PROG
  RAMMERINGSFIDUSDU IKKE VIL UD MED"
2 PRINT"KAN DU SKRIVEmmmmmmmmmmPO
```

```
KE774,0 HVOREFTERmmmmDIT PROGRAM IKKE  
KAN"  
3 PRINT"LISTES MEN NATURLIGVISSTADIG  
KØRES – VIL DU HAVE LISTEN IGEN"  
4 PRINT"BRUGER DU POKE774,26"  
5 POKE774,0:LIST
```

## LEKTION 24

### BEVÆGELSE

```
1 POKE36879,24:PRINT"hf1NIVEAU(1-9)?"  
2 GETA$:IFA$=""THEN2  
3 IFASC(A$) < 49ORASC(A$) > 57THEN2  
4 A=7657:B=38377:C=9:D=11:E=1:F=1:G=VAL(A$):P  
  OKE650,128  
5 PRINT"h":FORH=2TO7:FORI=1TO22:POKEA+H*2  
  2+I,250:POKEB+H*22+I,H:NEXT:NEXT  
6 IFC=0THENPOKE650,0:GOTO6  
7 H=8:I=2:IFRND(1) < .5THENI=3  
8 GETA$:POKEA+H*22+I,32:POKEA+D+506,32:H=H  
  +E:I=I+F  
9 IFPEEK(A+H*22+I)=250THENJ=J+(8-H)*G:GOSUB  
  19:E=-E  
10 IFH=22ANDD < > ITHENC=C-1:GOSUB19:GO  
  TO6  
11 IFA$="Z"THEND=D-1  
12 IFA$="M"THEND=D+1  
13 IFD=0THEND=1  
14 IFD=23THEND=22  
15 POKEA+H*22+I,81:POKEB+H*22+I,0:POKEA+D+50  
  6,120:POKEB+D+506,0  
16 IFH=1ORH=22THENE=-E  
17 IFI=1ORI=22THENF=-F  
18 FORK=1TO(9-G)*10:NEXT:GOTO8  
19 PRINT"hf1f9LIV"C"POINTS"J:RETURN
```

- A 7657
- B 38377
- C LIV

D BAT  
E BOLDENS LINJEFLAG  
F BOLDENS PLADSFLAG  
G NIVEAU  
H BOLDENS LINJE  
I BOLDENS PLADS  
J POINTS  
K PAUSE

1. Hvid skærm med sort kant. Niveau?
2. Tag imod niveau.
3. Korrekt indtastning?
4. Initialisering. Hvad gør POKEn? Prøv at udelade den og se, hvad der sker!
5. Lav mursten.
6. Hvis ikke flere liv, tilbage til normaltilstand. Uendelig løkke, billedet fryses (du skal bruge RUN STOP/RE-STORE for at komme ud).
7. Boldens startposition.
8. Check for indtastning. Slet gammel bold, gammelt bat. Position for ny bold.
9. Hvis mursten ramt, pointsforøgelse, skift boldens linje-flag.
10. Rammer bat ikke bold, – 1 liv, ny omgang.
11. Bat til venstre?
12. Bat til højre?
13. Bat for langt til venstre?
14. Bat for langt til højre?
15. Tegn og mal bold og bat.
16. Hvis bold rammer side, skift pladsflag.
17. Hvis bold rammer top eller bund, skift linje-flag.
18. Pause svarende til niveau.
19. Skriv liv tilbage, points.

Som overalt i bogen er programmerne det vigtigste. Det er



ved at analysere og forbedre dem, du lærer at programmere. Vær helt sikker på, at du har forstået til bunds, hvordan ovenstående fungerer.

Lav derefter programmer, hvor

1. Farverne er andre.
2. Spillet er lettere, fordi farten på bolden er mindre.
3. Spillet er lettere, fordi du ikke behøver at ramme så nøjagtigt. Prøv en formel af typen

IFABS(D-I) < 2 THEN

altså hvis bare ikke der rammes mere end én plads forskert, så . . .

4. Bolden ikke skifter linjeflag, hver gang den rammer en mursten.

```
1 POKE650,128:POKE36879,93:TI$="000000":A=9:B=
  11:C=50:PRINT"h"
2 PRINTTAB(A)"f1c+f4f9mmmmf1c+":IFRND(1) < .1T
  HENPRINTTAB(A+INT(RND(1)*5)+1)"nf3p"
3 IFD > 20ANDPEEK(7679+B) < > 160THEN3
4 POKEB+7679,90:POKEB+38399,2:IFRND(1) < .5THE
  NA=A-1
5 IFRND(1) < .5THENA=A+1
6 IFA=0THENA=1
7 IFA=16THENA=15
8 GETA$:IFA$="Z"THENB=B-1
9 IFA$="M"THENB=B+1
10 IFA$="1"THENC=50
11 IFA$="2"THENC=25
12 IFA$="3"THENC=0
13 D=D+1:IFD=1000THENPRINTTI$:END
14 FORE=1TOC:NEXT:GOTO2
```

A VEJ  
B BIL  
C GEAR  
D AFSTAND  
F PAUSE

1. Grøn skærm, start ur, initialiser, slet skærm.
2. Vej, modgående færdsel.
3. Ulykke?
4. Bil. Vej til venstre?
5. Vej til højre?
6. Vej for langt til venstre?
7. Vej for langt til højre?
8. Bil til venstre?
9. Bil til højre?
10. 1. gear?
11. 2. gear?
12. 3. gear?
13. Hvis strækning tilbagelagt, angiv tid.
14. Pause svarende til gear.

Prøv andre variationer.

NB! Prøv også

```
1 PRINT"h":POKE36879,8:A=9:B=9:C=200
2 POKEA*22+B+7657,32:IFRND(1) < .5THENA=A-1
3 IFRND(1) < .5THENA=A+1
4 IFRND(1) < .5THENB=B-1
5 IFRND(1) < .5THENB=B+1
6 IFA=7THENA=8
7 IFA=13THENA=12
8 IFB=7THENB=8
9 IFB=13THENB=12
10 POKEA*22+B+7657,87:POKE7887,43:GETA$:IFA$
```

```
      < > " " THEN 13
11  IFA=10 AND B=10 THEN 11
12  C=C-10
13  C=C-1:IF C < 0 THEN C=0
14  PRINT "hf2"C"vm":IF C = 0 THEN 14
15  GOTO 2
```

## LEKTION 25

### BETINGEDE UDTRYK

```
1 POKE650,128:POKE36879,25:A=10:B=10:PRINT"h"
2 POKEA*22+B+7657,160:POKEA*22+B+38377,C:GE
  TA$:IFA$="W"THENA=A-1
3 IFA$="Z"THENA=A+1
4 IFA$="A"THENB=B-1
5 IFA$="S"THENB=B+1
6 IFA$="F"THENGOSUB12
7 IFA=0THENA=1
8 IFA=24THENA=23
9 IFB=0THENB=1
10 IFB=23THENB=22
11 GOTO2
12 GETA$:IFA$=" "THEN12
13 IFASC(A$) < 49ORASC(A$) > 56THEN12
14 C=VAL(A$)-1:RETURN
```



- A LINJE
- B PLADS
- C FARVE

1. Hvid skærm, prikkens startposition, slet skærm.
2. Tegn og mal prik, check for indtastning, prik op?
3. Prik ned?
4. Prik til venstre?
5. Prik til højre?
6. Farveskift?
7. Prik for langt oppe?
8. Prik for langt nede?
9. Prik for langt til venstre?
10. Prik for langt til højre?
11. Om igen.
12. Check for indtastning af farve.
13. Korrekt indtastning?
14. Farveskift.

Dette program er egentlig på 23 linjer. Linje 1 er således f.eks. sammensat af 5 linjer. Denne sammenpresning foretager vi givetvis af hensyn til den anvendte hukommelse, men dog i langt højere grad på grund af den større overskuelighed. Vi må kunne overse et program for at gøre det så godt som muligt. Idealet bliver det kompakte, hurtige, effektive, overskuelige og letforståelige program. Jo mere vi kan presse linjer, der hjælper hinanden med at udføre bestemte opgaver, sammen til faste, omflyttelige »klodser«, jo mere kan vi koncentrere os om de store linjer i vores program. Det synes måske krukke i programmer af denne størrelse, men forestil dig et program af tidobbelt længde. Så er det rart, hvis vi kan finde initialisationen og ikke løber vild i et virvar af overflødige GOTOer og linjenumre.

Betragter vi nu vores program med disse øjne, må det slå os, at der er noget påfaldende »uforkortet« ved linjerne 3-10. Men hvad skal man gøre, det er IF-linjer, som man ikke uden videre kan hægte sammen. Her kommer imidlertid et andet fænomen os til hjælp: SANDHEDSVÆRDIEN og DET BETINGEDE UDTRYK.

Hvad betyder egentlig

IFA=1 THEN 666

Jo, maskinen skal gå til linje 666, hvis og kun hvis det er opfyldt eller SANDT, at  $A=1$ . Men hvordan ved maskinen nu det? Jo, den tilkender nemlig helt automatisk enhver relation et tal, afhængigt af, om relationen er sand eller falsk, en såkaldt SANDHEDSVÆRDI, nemlig

-1 (minus 1) hvis relationen er sand

og

0 (nul) hvis relationen er falsk

Det kan du selv kontrollere. Skriv

? "2+2" (skriv strengen "2+2")

derefter

? 2+2 (skriv resultatet af 2+2)

og endelig

? 2+2=4 (skriv SANDHEDSVÆRDIEN af relationen  $2+2=4$ )

Er du med? Prøv nu

? 2+2=5

Eller

?2+2=4AND2+2=5

Forklar det sidste resultat. Lad os nu vende tilbage til vores program. Tag f.eks.

IFA\$="Z"THEN A=A+1

Hvad, om vi i stedet skrev.

A=A-(A\$="Z")

Ja, hvis A\$="Z", så er sandhedsværdien af relationen -1, og vi får

A=A-(-1)

eller simpelt hen

A=A+1

Hvis derimod A\$ ikke er "Z", bliver sandhedsværdien 0, og vi får

A=A-(0)

Pengene passer. Og dermed er de 23 linjer, der blev til 14, faktisk blevet til SEKS! Nemlig

```
1 POKE650,128:POKE36879,25:A=10:B=10:PRINT"h"
2 POKEA*22+B+7657,160:POKEA*22+B+38377,C:GET
  A$:IFA$="F"THENGOSUB4
3 A=A+(A$="W")-(A$="Z"):B=B+(A$="A")-(A$="S"):A
  =A+(A=24)-(A=0):B=B+(B=23)-(B=0):GOTO2
4 GETA$:IFA$=""THEN4
```

```

5 IFASC(A$)<49ORASC(A$)>56THEN4
6 C=VAL(A$)-1:RETURN

```

Tryllekunsten består selvfølgelig i at blive af med IFerne ved at »indbygge« betingelsen i relationen eller udtrykket. Et udtryk af typen

```
A=A-(A$="Z")
```

kaldes derfor også et BETINGET UDTRYK.

NB! Prøv

```

1 POKE650,128:POKE36879,8:TI$="000000":A=1000:B
  =2100
2 PRINT"h":FORC=8164TO8185:POKEC,160:NEXT
3 PRINT"hf2TID"INT(TI/60):PRINT"FUEL"INT(A)"vm"
  :PRINT"JET"INT(D):PRINT"FART"INT(E)"vm"
4 PRINT"HØJDE"INT(B)"vm":IFB=0ANDE>=-1THE
  NPRINT"BLØD LANDING":POKE650,0:END
5 IFA<=0ORB=0ANDE<-1THENPRINT"NEDSTYR
  TNING":POKE650,0:END
6 GETA$:D=D+(D>0)*.5-(A$<>"")ANDD<9):A=A
  -D/10:E=E+D/50-.05
7 POKE(20-INT((B-1)/100))*22+7694,32:B=B+E:IFB<
  0THENB=0
8 IFB>2100THENB=2100
9 POKE(20-INT((B-1)/100))*22+7694,1:GOTO3

```



# LEKTION 26

## SKÆRMKODER

```
1 A=7657:B=38377:C=2:D=2:E=22:F=20:POKE650,12
  8:POKE36879,25:PRINT"h"
2 FORG=2TO22:FORH=2TO20STEP2:POKEA+G*22+
  H,46:POKEB+G*22+H,5:NEXT:NEXT
3 FORG=1TO21STEP2:FORH=1TO23:POKEA+G+H*
  22,160:POKEB+G+H*22,6:NEXT:IFG=1ORG=21TH
  EN5
4 FORI=1TO7:J=G+(INT(RND(1)*21)+2)*22:POKEA+J,
  46:POKEB+J,5:NEXT
5 NEXT:FORG=1TO23STEP22:FORH=1TO21:POKEA
  +G*22+H,160:POKEB+G*22+H,6:NEXT:NEXT
6 POKEA+C*22+D,32:GETA$
7 C=C+(A$="W"ANDPEEK(A+(C-1)*22+D) < 128)-
  (A$="Z"ANDPEEK(A+(C+1)*22+D) < 128)
8 D=D+(A$="A"ANDPEEK(A+C*22+D-1) < 128)-(A$
  ="S"ANDPEEK(A+C*22+D+1) < 128)
9 IFPEEK(A+C*22+D)=46THENK=K+10
10 POKEA+C*22+D,90:POKEB+C*22+D,2:POKEA+E
  *22+F,46:POKEB+E*22+F,2
11 E=E+((C < EORRND(1) < .5)ANDPEEK(A+(E-1)*22+
  F) < 128)
12 E=E-((C > EORRND(1) < .5)ANDPEEK(A+(E+1)*22+
  F) < 128)
13 F=F+((F > DORRND(1) < .5)ANDPEEK(A+E*22+F-1)
  < 128)
14 F=F-((F < DORRND(1) < .5)ANDPEEK(A+E*22+F+1)
  < 128):POKEA+E*22+F,88:POKEB+E*22+F,0
15 IFC=EANDD=FTHEN15
16 PRINT"hf7f9"K:GOTO6
```

A 7657  
B 38377  
C DIN LINJE  
D DIN PLADS  
E UHYRETS LINJE  
F UHYRETS PLADS  
G KONTROLVARIABEL  
H KONTROLVARIABEL  
I KONTROLVARIABEL  
J KONTROLVARIABEL  
K POINTS

1. Initialisation, hvid skærm, slet skærm.
2. Tegn og mal prikker.
3. Tegn og mal labyrint.
4. Tegn og mal passager.
5. Tegn og mal ydermur.
6. Slet dig, check for indtastning.
7. Du op eller ned?
8. Du til venstre eller højre?
9. Hvis prik spist, pointsforøgelse.
10. Tegn og mal dig, slet uhyre.
11. Uhyre op?
12. Uhyre ned?
13. Uhyre til venstre?
14. Uhyre til højre? Tegn og mal uhyre.
15. Fangede uhyret dig?
16. Skriv points.

Vi har her næsten et »rigtigt« spil. Du kan styre din »brik« gennem labyrinten og spise de små grønne prikker, men ikke gå igennem mure. Hver spist prik giver ti points, og du kan hele tiden se i øverste venstre hjørne, hvor mange du har fået. Samtidig skal du selvfølgelig undgå uhyret, der jager dig, og samtidig efterlader et spor af røde prikker, der

også giver points, hvis du spiser dem. I denne version giver de også ti points. Lav en version, hvor de giver tyve (peek prikkens farve). Eller lad små hjerter dukke op hist og her, som f.eks. giver 100 points. Eller lav endnu bedre versioner.

*Lidt kedeligt* er det nu immervæk. Du burde sige noget (altså din »brik«), mens du farer igennem labyrinten, og uhyret noget andet, mere faretruende. En prik skulle også give et lille gisp fra sig, når den mødte sit endeligt, et hjerte kunne give en fanfare, og dit uundgåelige endeligt en lille sørgelig melodistump. Så ville det straks begynde at ligne noget!

Faktisk skal vi beskæftige os med lyd i hele resten af denne del. Fjerde har vi så reserveret til de lidt mere avancerede ting, definering af egne karakterer (rigtige *tegnefilmsuhyrer*!) bl.a. Før vi fanger an med noget helt nyt, er der dog som sædvanlig lige et par småting at samle op! Det bruger vi resten af kapitlet til!

Læg mærke til, hvordan vi fik labyrinten, nemlig med 160, altså  $32+128$ , altså  $32 =$  mellemrum omvendt. Et blik på skærmmkodetabellen i appendikset bag i bogen vil vise dig, at den højeste skærmmkode er 127, resten er »omvendte«.

Men sæt, vi nu vil bruge det andet karaktersæt, det med de små bogstaver? Prøv, mens du spiller labyrintspillet, at taste COMMODORE/SHIFT. Den røde rude skifter til et Z, »uhyret« til et X. Så nemt er det! Du kan naturligvis også POKE dig til skiftet. Så betyder

POKE36869,242 skift til karaktersæt 2

og

POKE36869,240 skift (tilbage) til karaktersæt 1

Du kan altså ikke have karakterer fra de to sæt på skærmen

på én gang, ganske uanset, om du bruger tastaturet eller POKE.

Skriv

?PEEK(214)

Maskinen oplyser, på hvilken linje markøren befandt sig, da du »peekede«. Skriv nu

POKE214,100

Nu skal markøren være i linje 100, og det er selvfølgelig umuligt. Maskinen reagerer da også noget besynderligt. Prøv en LIST. Den er nervesammenbruddet nær.

Faktisk *kan* en datamat godt få nervesammenbrud. Re-  
kreationsopholdet er dog kortvarigt og billigt. Sluk for ma-  
skinen og tænd igen, så er den i bogstavelig forstand så god  
som ny.

Symptomerne er lidt forskellige. Fælles for dem alle er  
det dog, at maskinen ikke længere reagerer fornuftigt på  
fornuftige ordrer, eventuelt slet ikke. Men nogen uhelbre-  
delig sygdom er det altså ikke just.

NB! Prøv

```
1 REM HVIS DU UDSKIFTER 9 I 35 MED ET LA  
  VERE TAL, HAR DU EN BEDRE MODSTANDER  
2 POKE36879,24:FORA=1TO6:READA$(A):NEXT  
3 A=100:B=100:C=1000:D=100:E=100:F=1000:G=9:  
  H=9:I=9:J=9  
4 K=1:L=L+1:PRINT"hf1f9"L"v. DAG":IFMTHENPRI  
  NT"f9FJENDENS SVAR:":PRINT"f9"A$(M)  
5 PRINT:PRINT"f7f9PATRIA"  
6 PRINTA"FLY (PROD"G"v)":PRINTB"SKIBE (PROD  
  "H"v)":PRINTC"SOLDATER"
```

```

7 PRINT:PRINT"f3f9HOSTILIA"
8 PRINTD"FLY (PROD"I"v)":PRINTE"SKIBE (PROD"
  J"v)":PRINTF"SOLDATER"
9 PRINT:PRINT"f51. BOMB LUFTHAVN":PRINT"2.
  BOMB HAVN":PRINT"3. SØSLAG"
10 PRINT"4. INVASION":PRINT"5. BOMB FLYFABR
  IK":PRINT"6. BOMB DOK"
11 PRINT"7. INGEN ORDRE":PRINT"f9ORDRE?"
12 GETA$:IFA$="7"THEN21
13 IFA$ < "1"ORA$ > "6"OR(A$ < "3"ORA$ > "4")AN
  DA=0ORA$="3"ANDB=0ORA$="4"ANDC=0THE
  N12
14 ONVAL(A$)GOTO15,16,17,18,19,20
15 N=A:O=D:GOSUB35:A=N:D=O:ONKGOTO21,29
16 N=A:O=E:GOSUB35:A=N:E=O:GOTO21
17 N=B:O=E:GOSUB35:B=N:E=O:ONKGOTO21,29
18 N=C:O=F:GOSUB35:C=N:F=O:ONKGOTO21,29
19 N=A:O=I:GOSUB35:A=N:I=O:GOTO21
20 N=A:O=J:GOSUB35:A=N:J=O
21 P=0
22 P=P+1:IFP=25THEN31
23 M=INT(RND(1)*6)+1:IFM=1AND(A=0ORD=0)ORM
  =2AND(B=0ORD=0)ORM=3AND(B=0ORE=0)THE
  N22
24 IFM=4AND(C=0ORF=0)ORM=5AND(D=0ORG=0)
  ORM=6AND(D=0ORH=0)THEN22
25 K=2:ONMGOTO15,26,17,18,27,28
26 N=B:O=D:GOSUB35:B=N:D=O:GOTO29
27 N=G:O=D:GOSUB35:G=N:D=O:GOTO29
28 N=H:O=D:GOSUB35:H=N:D=O
29 A=A+G:B=B+H:D=D+I:E=E+J:IFL < 25THENG=G
  -(G < 9):H=H-(H < 9):I=I-(I < 9):J=J-(J < 9)
30 GOTO4
31 G=A+B+C:H=D+E+F:PRINT"f3f9";:IFG=HTHENPR
  INT"FRED";

```

```

32 IFG > H THEN PRINT "PATRIA VANDT";
33 IFG < H THEN PRINT "HOSTILIA VANDT";
34 GOTO 34
35 N = N - INT(RND(1) * O / 9): IF N < 0 THEN N = 0
36 O = O - INT(RND(1) * N / 10): IF O < 0 THEN O = 0
37 RETURN
38 DATA LUFTHAVN BOMBET, HAVN BOMBET, SØS
    LAG, INVASION, FLYFABRIK BOMBET, DOK BOM
    BET

```

## LEKTION 27

### LYDEFFEKTER

Der er fire LYDGENERATORER i VIC, det vil sige, VIC læser, hvilken tone den skal spille, i fire adresser. Disse er

36874 BAS

36875 ALT

36876 SOPRAN

36877 STØJ

Hver af disse adresser kan naturligvis indeholde en værdi fra 0 til 255, men kun værdier på 128 eller derover resulterer i en lyd. Endelig kan lydstyrken sættes fra 1 til 15 i adresse 36878. Prøv

- 1 POKE36878,15:FORA=36874TO36877:FORB=128TO255:POKEA,B:PRINT"POKE"A","B
- 2 FORC=1TO500:NEXT:POKEA,0:FORC=1TO100:NEXT:NEXT:NEXT

Læg mærke til, at 255 i alle fire generatorer er en meget dyb tone. Du vil måske umiddelbart have svært ved at opfatte det, 36877 genererer, som toner, i det hele taget er det vanskeligt at se, hvad nytte man skulle kunne have af denne »støj«. Faktisk kan den, eventuelt kombineret med toner, give udmærkede lydeffekter. Lad os prøve med nogle af vores spil:

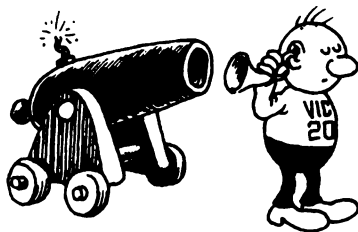
- 1 A=7657:B=38377:C=2:D=2:E=22:F=20:POKE650,128:POKE36878,15:POKE36879,25:PRINT"h"
- 2 FORG=2TO22:FORH=2TO20STEP2:POKE36874,G\*

```

5+128:POKE36875,H*5+128
3 POKEA+G*22+H,46:POKEB+G*22+H,5:NEXT:NEXT
4 FORG=1TO21STEP2:FORH=1TO23:POKE36874,G*
  5+128:POKE36875,H*5+128
5 POKEA+G+H*22,160:POKEB+G+H*22,6:NEXT:IF
  G=1ORG=21THEN7
6 FORI=1TO7:J=G+(INT(RND(1)*21)+2)*22:POKEA+J
  ,46:POKEB+J,5:NEXT
7 NEXT:FORG=1TO23STEP22:FORH=1TO21:POKEA
  +G*22+H,160:POKEB+G*22+H,6:NEXT:NEXT
8 POKE36875,0
9 POKE36874,0:POKE36877,0:POKEA+C*22+D,32:GE
  TA$:IFA$ < > ""THENPOKE36877,200:POKE36877
  ,0
10 C=C+(A$="W"ANDPEEK(A+(C-1)*22+D) < 128)-(A
  $="Z"ANDPEEK(A+(C+1)*22+D) < 128)
11 D=D+(A$="A"ANDPEEK(A+C*22+D-1) < 128)-(A$
  ="S"ANDPEEK(A+C*22+D+1) < 128)
12 IFPEEK(A+C*22+D)=46THENK=K+10:POKE36874,
  250
13 POKEA+C*22+D,90:POKEB+C*22+D,2:POKE36874,
  0:POKEA+E*22+F,46:POKEB+E*22+F,2
14 E=E+((C < EORRND(1) < .5)ANDPEEK(A+(E-1)*22
  +F) < 128)
15 E=E-((C > EORRND(1) < .5)ANDPEEK(A+(E+1)*22
  +F) < 128)
16 F=F+((F > DORRND(1) < .5)ANDPEEK(A+E*22+F-1)
  < 128)
17 F=F-((F < DORRND(1) < .5)ANDPEEK(A+E*22+F+1)
  < 128)
18 POKE36874,200:POKEA+E*22+F,88:POKEB+E*22+
  F,0
19 IFC=EANDD=FTHENFORA=254TO127STEP-1:PO
  KE36874,A:FORB=1TO50:NEXT:NEXT:GOTO21
20 PRINT"hf7f9"K:GOTO9

```





## 21 GOTO21

1. Lydstyrke 15.
2. »Musik« til prikkerne.
4. »Musik« til labyrinten.
8. Sluk for musikken.
9. Sluk for din lyd. Hvis du »går«, »trin«.
12. Hvis prik spist, dyt.
13. Sluk for uhyrets lyd.
18. Lyd, når uhyret går.
19. »Melodi«, når du bliver spist.

Læg mærke til »musikken«, når labyrinten tegnes, og hvor enkelt den uddrages af kontrolvariabler G og H. Husk også, at der skal slukkes. RUN STOP/RESTORE gør dog kål på al musik, idet den sætter alle fem bytes = 0.

Læg også mærke til »trinene«. Alt efter, hvordan 36877s »susen« stiger, falder i tonehøjde eller lydstyrke eller kortes af, kan den bruges som skud, stormvejr, eksplosioner, jetmotorer m.m.m.

Som du ser, er der draget nytte af, at der er flere lydgeneratorer eller »stemmer«, som kan lyde samtidig. Det får vi selvfølgelig mere brug for, når vi skal til at lave »rigtig« musik på VIC.

```
1 POKE650,128:POKE36878,15:POKE36879,25:A
  =10:B=10:PRINT"h"
```

```

2 POKE36874,A*5+128:POKE36875,B*5+128:POKE368
  76,C*18+128
3 POKEA*22+B+7657,160:POKEA*22+B+38377,C:GET
  A$:IFA$="F"THENGOSUB5
4 A=A+(A$="W")-(A$="Z"):B=B+(A$="A")-(A$="S"):A
  =A+(A=24)-(A=0):B=B+(B=23)-(B=0):GOTO2
5 GETA$:IFA$=" "THEN5
6 IFASC(A$)<49ORASC(A$)>56THEN5
7 C=VAL(A$)-1:RETURN

```

Prøv på samme måde, hvad du kan gøre for andre af vores spil. Automobilsillet skal selvfølgelig have en motorstøj, der stiger med gearet. De modgående »cykler« kan få en cykelklokke, og så må vi selvfølgelig sørge for et solidt brag, når spillet slutter + en ambulance for de blodtørstige.

Bolden og murstenene kan naturligvis også kun blive bedre. Men tekstspillene kan også blive mere spændende. Hvorfor skilte med tunge skridt og løbende fødder, når du kan *høre* dem?! Bliver lyden af dem kraftigere eller svagere?

I rumskibssillet kan du høre alt fra affyringen af laser og faser over fuldtræffere og bommere til kraftskjoldets effekt og skibets energiniveau. Klem et par blinkende lamper på, og du befinder dig næsten på en rigtig kommandobro. Skal simulationen være så nøjagtig, vil det måske være en god idé at indføre »rigtig tid«, så det går ud over din kampdygtighed, hvis du er for længe om at beslutte dig for en ordre. En strengdeling kunne muliggøre en mere realistisk ordregivning.

– Ordre, kaptajn Kuk?

– Fuld kraft frem, Mr. Silly!

Og så er der da i øvrigt ingen, der tror på, at din forvoksede skrivemaskine er en datamat, hvis ikke den opfører sig ligesom i tegneserierne:

```

1 POKE36878,15:PRINT"h2+2="";FORA=1TO5000

```

```

:NEXT
2 FORA=1TO500:B=INT(RND(1)*128)+128:POKE36874
,B:POKE36879,B:NEXT:PRINT5:POKE36874,0

```

NB! Prøv

```

1 POKE36878,15:POKE36879,25:A=7932:B=38652:C=
160:PRINT"h"
2 D=INT(RND(1)*11):E=INT(RND(1)*11):F=INT(RND(1)
*8)
3 POKE36874,F+247:POKE36877,255:POKE36877,0:D
=D*22
4 POKEA-D-E,C:POKEB-D-E,F:POKEA-D+E,C:POK
EB-D+E,F
5 POKEA+D-E,C:POKEB+D-E,F:POKEA+D+E,C:POK
EB+D+E,F:D=D/22:E=E*22
6 POKEA-D-E,C:POKEB-D-E,F:POKEA+D-E,C:POK
EB+D-E,F
7 POKEA-D+E,C:POKEB-D+E,F:POKEA+D+E,C:POK
EB+D+E,F
8 IFRND(1)<.005THENPOKE36874,0:PRINT"h":FORD
=1TO2000:NEXT
9 GOTO2

```

# LEKTION 28

## TONER

Vi vil i tredje dels tre sidste lektioner undersøge VICs muligheder for musikalsk udfoldelse. Dette er naturligvis et stort område, og vi skal da også nøjes med at lade disse få sider fungere som en igangsætter, som det har været politikken bogen igennem.

Vores opmærksomhed koncentrerer sig nu naturligt om de værdier i adresserne 36874, 36875 og 36876, der giver os hel- eller halvtoner. Det er

135 C  
143 CIS  
147 D  
151 DIS  
159 E  
163 F  
167 FIS  
175 G  
179 GIS  
183 A  
187 AIS  
191 H  
195 C  
199 CIS  
201 D  
203 DIS  
207 E  
209 F  
212 FIS  
215 G

217 GIS  
219 A  
221 AIS  
223 H  
225 C  
227 CIS  
228 D  
229 DIS  
231 E  
232 F  
233 FIS  
235 G  
236 GIS  
237 A  
238 AIS  
239 H  
240 C  
241 CIS

Hver stemme spænder altså over godt tre oktaver. De tre stemmer er imidlertid forskudt en oktav i forhold til hinanden, så at den totale spændvidde bliver fem oktaver.

POKE36874,225

giver altså den samme tone som

POKE36875,195

og

POKE36876,135

Her er hele skalaen:

```

1 POKE36878,15:FORA=36874TO36875
2 FORB=1TO12:READC:POKEA,C:FORD=1TO1000:N
  EXT:NEXT:POKEA,0:RESTORE:NEXT
3 FORA=1TO38:READB:POKE36876,B:FORD=1TO100
  0:NEXT:NEXT:POKE36876,0
4 DATA135,143,147,151,159,163,167,175,179,183,187,1
  91,195,199,201,203,207,209,212,215
5 DATA217,219,221,223,225,227,228,229,231,232,233,2
  35,236,237,238,239,240,241

```

Disse toner er naturligvis ikke *absolut* rene. Det vil kræve »frekvensmodulation«, dvs. du skaber en ny tone ved at skifte meget hurtigt frem og tilbage mellem to andre. Dette falder imidlertid uden for denne bogs sigte, idet den så skulle handle om VIC som musikinstrument eksklusivt. Da vi her skal snakke om VIC generelt, må vi nøjes med, hvad der kan klares på tre lektioner.

Vi kan selvfølgelig også få tonerne direkte fra tastaturet. En meget simpel udgave er

```

1 POKE36878,15:POKE36879,8:FORA=1TO8:REA
  DA(A):NEXT:PRINT"hf2MUSIK"
2 GETA$:IFA$=" "THEN2
3 IFASC(A$)>48ANDASC(A$)<57THENA=VAL(A$):
  POKE36874,A(A):POKE36879,A*17-9
4 FORA=1TO100:NEXT:POKE36874,0:GOTO2
5 DATA225,228,231,232,235,237,239,240

```

Hvor sindrigt et sådant program ellers kan blive, er og forbliver VICs tastatur et tastatur og intet klaviatur. Mulighederne ligger altså hovedsagelig i at *programmere* maskinen til at spille et stykke musik. Vi vil se, hvor langt vi kan komme med dette i lektion 29 og 30. Lad os prøve en simpel melodi for at finde frem til principperne:

```

1 POKE36878,15
2 FORA=1TO13:READB:POKE36874,B:FORC=1TO250:
  NEXT:POKE36874,0:FORC=1TO5:NEXT:NEXT
3 DATA225,225,225,231,228,228,228,232,231,231,228,
  228,225

```

Lidt mere kompliceret bliver det, når tonerne har forskellig længde:

```

1 POKE36878,15
2 FORA=1TO23:READB:READC:POKE36874,B:FORD=
  1TOC:NEXT:POKE36874,0:FORD=1TO5:NEXT:NE
  XT
3 DATA215,200,219,200,223,200,225,200,228,800,231,2
  00,231,200,235,200,231,200,228,800
4 DATA225,200,225,200,225,200,225,200,223,200,223,2
  00,223,200,223,200,219,200,223,200
5 DATA225,200,219,200,215,800

```

Læg mærke til, at DATA-linjerne indeholder både toner og tonelængde.

I næste lektion skal vi spille med flere stemmer.

**NB! Prøv**

```

1 DIMA$(49):FORA=1TO49:READA$(A):NEXT
2 A$=A$(INT(RND(1)*49)+1):PRINT"FIND MIT OR
  D (10mmmmmmBOGSTAVER). INDTASTmmmF
  ORSØG."
3 PRINT"RIGTIGE BOGSTAVERmmmmmmANGIVES
  ; RIGTIGTmmmmmmPLACEREDE MED OMVE
  NDT SKRIFT."
4 B=B+1:PRINT:PRINT"f6"B;:INPUTB$:PRINTTAB((B
  < 10)+6)"nf3";:IFLEN(B$) < > 10THEN4
5 FORA=1TO10:PRINT"-";:FORC=1TO10:IFMID$(B$

```

```

    ,A,1)=MID$(A$,C,1)THENPRINT"v"MID$(B$,A,1);
6 NEXT:IFMID$(A$,A,1)=MID$(B$,A,1)THENPRINT"v
  f9"MID$(A$,A,1)"f0";
7 NEXT:PRINT
8 IFA$=B$THEN8
9 GOTO4
10 DATAABBREVIERE,BABYLONIER,CALVINISME,
  DAGARBEJDE,ECHAUFFERE,FABELAGTIG,GAFF
  ELDELT
11 DATAHABILITERE,IDEALISERE,JAKOBSKAMP,K
  ABALEKORT,LABANALDER,MADAPOLAME,NAB
  OVINKEL
12 DATAOBDUCERING,PACIFICERE,RABBINISME,S
  ADELGJORD,TABERNAKEL,UAF LADELIG,VAFF
  ELJERN
13 DATAXYLOFONIST,YDERLIGGER,ZEBRAFINKE,
  ØJENSMINKE,AVNEKNIPPE,BØTTEPAPIR,CYLIN
  DRISK
14 DATADØTRESKOLE,EXCENTRISK,FØRSTEMAND
  ,GØGLERVOGN,HØVEDSMAND,ISTIDSSAND,JUV
  ELSKRIN
15 DATAKØTERAGTIG,LØSSLUPPEN,MØRKRADE
  T,NØGLEVIDDE,OVERVINTRE,PØLSEMAGER,RØ
  VERBANDE
16 DATASØSTERKAGE,TØVEJRSDAG,UVURDERLIG,
  VULNERABEL,YNKELIGHED,ZINKSTØBER,ØSTN
  ORDISK

```



# LEKTION 29

## MELODIER

```
1 POKE36878,15:FORA=1TO32:READB:READC:READ
  D:POKE36874,B:POKE36875,C:FORE=1TOD:NEXT:N
  EXT
2 POKE36874,0:POKE36875,0
3 DATA225,235,250,0,0,125,235,237,125,231,235,250,2
  35,232,250,225,231,250,235,232,250
4 DATA231,235,250,235,235,250,223,228,250,235,231,2
  50,228,232,250,225,232,250
5 DATA225,231,250,235,232,250,231,235,250,235,235,2
  50,225,235,250,0,0,125,235,237,125
6 DATA231,235,250,235,232,250,225,231,250,235,232,2
  50,231,235,250,235,235,250
7 DATA223,228,250,235,228,250,228,235,250,235,235,2
  50,225,231,500,225,225,500
```

A KONTROLVARIABEL  
B BAS  
C DISKANT  
D TONELÆNGDE  
E KONTROLVARIABEL

1. Lydstyrke 15, læs bas, diskant og tonelængde, spil bas og diskant, hold tonerne.
2. Sluk bas og diskant.
- 3-7. Melodi.

Lad os se på de første 12 data:

225,235,250

## C BAS, G DISKANT, KVARTNODER

0,0,125

## OTTENDEDELSPAUSE I BAS OG DISKANT

235,237,125

## G BAS, A DISKANT, OTTENDEDELSNODER

231,235,250

## E BAS, G DISKANT, KVARTNODER

Er det svært? Måske i begyndelsen, men med en smule træning kan du let transskribere direkte fra nodebladet ind i datasætningerne. Og så er det næsten ubegrænset, hvad du kan få ud af musikfunktionen. Vi kunne naturligvis også have »talt« 200 på en kvartnode, i stedet for 250. Det ville have givet et hurtigere *tempo*. De fleste noder er forsynet med tempoangivelse, som du så kan oversætte til en kontrolvariabel på din pauseløkke.

Lad os engang prøve tre stemmer:

```
1 POKE36878,15:FORA=1TO27:READB:READC:READ
  D:READE:POKE36874,B:POKE36875,C:POKE36876,D
2 FORF=1TOE:NEXT:POKE36874,0:POKE36875,0:POK
  E36876,0:FORF=1TO5:NEXT:NEXT
3 DATA225,215,195,375,,,195,375,225,215,195,250,,,201
  ,125,,,207,375,225,215,207,250
4 DATA,,201,125,,,207,250,,,209,125,225,215,215,750,22
  5,215,225,125,,,225,125,,,225,125
5 DATA,,215,125,,,215,125,,,215,125,225,215,207,125,,,
  207,125,,,207,125,,,195,125
```

6 DATA,,195,125,,,195,125,215,215,215,250,,,209,125,,,  
207,250,,,201,125,225,215,195,750

A KONTROLVARIABEL

B NEDRE BAS

C ØVRE BAS

D DISKANT

E TONELÆNGDE

F KONTROLVARIABEL

1. Lydstyrke 15, læs nedre og øvre bas, diskant og tonelængde, spil nedre og øvre bas og diskant.
  2. Hold tonerne, sluk nedre og øvre bas og diskant, .005 sekunders pause mellem hver tone.
- 3-6. Melodi.

Sikke mange kommaer! Det er, fordi vi underforstår alle nullerne. Kan man det? Ja, et komma i en DATA-linje læses blot som en nul-værdi, så det kan vi sagtens. F.eks. betyder altså

195,375,,,195

præcis det samme, som hvis der havde stået

195,375,0,0,195

Læg også mærke til, at rytmen er en kende mere avanceret. Og hvorfor ikke? En kontrolvariabel kan antage en hvilken som helst grænseværdi.

Vi kan endda spille »piano« eller »forte«, pianissimo eller fortissimo, vi har faktisk ikke mindre end femten forskellige lydstyrker at vælge imellem. Så måske skulle vi så småt overveje at spille musik på VIC?

NB! Prøv

```
1 PRINT"hf2PHANTASIE":POKE36878,15:POKE36879,
  8:DIMA(16):FORA=1TO16:READA(A):NEXT
2 A=INT(RND(1)*16+1):FORB=36874TO36876:POKEB,
  A(A):FORC=1TO100:NEXT:POKEB,0:NEXT
3 POKE36877,245:POKE36877,0:POKE36879,INT(RND
  (1)*8)*17+8
4 PRINTMID$("f1f2f3f4f5f6f7f8",INT(RND(1)*8)+1,1)"Q
  ";;GOTO2
5 DATA195,201,207,209,215,219,223,225,228,231,232,2
  35,237,239,240,
```

## LEKTION 30

### MUSIK

```
1 POKE36878,15:POKE36879,8:PRINT"h":FORA=1TO3
  0:READB:READC:READD:READE
2 POKE36874,B:POKE36875,C:POKE36876,D:FORF=1T
  OE:NEXT:NEXT:POKE36874,0:POKE36875,0
3 GOTO3
4 DATA,219,219,75,,217,217,75,,219,219,1200,,300,,215
  ,215,150,,209,209,150
5 DATA,207,207,150,,201,201,150,,199,199,300,,201,201
  ,600,,1800,219,219,75
6 DATA217,217,,75,219,219,,1200,,300,207,207,,300,20
  9,209,,300,199,199,,300
7 DATA201,201,,600,,1800,183,183,,75,179,179,,75,183
  ,183,,1200,,300,175,175,,150
8 DATA163,163,,150,159,159,,150,147,147,,150,143,143,
  ,300,147,147,,600
```

Skriv selv resten! Nåja, mindre kan vel også gøre det.

Billede og lyd kombineret giver selvfølgelig nogle ganske særlige muligheder. F.eks. kan du lade teksten til en sang, du vil lære, komme frem på skærmen, mens maskinen spiller melodien:

```
1 POKE36878,15:PRINT"h":POKE36879,10:DIMA$(31)
  :A$="f2f3f4f5f6f7f8"
2 A=A+1:READA$(A):IFA < 8THEN2
3 RESTORE
4 A=A+1:READA$(A):IFA < 24THEN4
5 RESTORE:READB$
6 A=A+1:READA$(A):IFA < 31THEN6
```

```

7 FORA=1TO8:READB$:NEXT:FORA=1TO31:READB
:READC:POKE36874,B:POKE36875,B:POKE36876,B
8 B$=MID$(A$,INT(RND(1)*7)+1,1):PRINTTAB(INT(R
ND(1)*17))"f9"B$A$(A):PRINT
9 FORD=1TOC:NEXT:POKE36874,0:POKE36875,0:PO
KE36876,0:FORD=1TO5:NEXT:NEXT
10 FORA=1TO1000:NEXT:RUN
11 DATAAND,WHEN,THE,SAINTS,GO,MARCH,ING,
IN,I,WANT,TO,BE,IN,THE,NUM,BERS
12 DATA225,200,231,200,232,200,235,1000,225,200,23
1,200,232,200,235,1000,225,200
13 DATA231,200,232,200,235,400,231,400,225,400,231,
400,228,1000,231,200,231,200,228,200
14 DATA225,800,231,400,235,400,235,200,232,1000,23
1,200,232,200,235,400,231,400,225,400
15 DATA228,400,225,1000

```

A\$(31) TEKST

A\$ FARVER

B\$ FARVE

A KONTROLVARIABEL

B TONE

C TONELÆNGDE

D PAUSE

1. Lydstyrke 15, slet skærm, sort skærm med rød kant, skaf plads til tekst, farver.
2. Læs tekst ind i sæt.
3. Tilbage.
4. Læs tekst ind i sæt.
5. Tilbage, spring AND over.
6. Læs tekst ind i sæt.
7. Spring resten af teksten over, læs tone og tonelængde, spil tone i alt, tenor og sopran.
8. Skriv tekst med omvendt skrift og tilfældig farve på et

- tilfældigt sted på linjen og spring en linje over.
9. Hold tonen, sluk alt, tenor og sopran, .005 sekunders pause mellem hver tone.
  10. 1 sekunds pause, om igen.
  11. Tekst.
  - 12-15. Melodi.

Læg mærke til, hvordan RESTORE »sparer« på vores ordforråd. Kunne man gøre noget lignende med melodien?

Prøv at lave noget sjov grafik, der danser i takt med musikken og eventuelt angiver tonehøjde!

De to eksempler er selvfølgelig for så vidt modsætninger. De viser, på hvor forskellig vis, maskinens musikalske funktion kan anvendes. Mine forudsætninger ud i det musikalske er (mildt sagt) minimale, men i det første eksempel er der dog gjort forsøg på at omsætte et nodebillede til noget, maskinen kan spille. Andre vil naturligvis komme meget bedre af sted med dette end jeg.



I det andet eksempel har vi brugt lutter »pseudo-akkorder«. Det er simpelt hen den samme tone, vi spiller i de tre stemmer. Det er denne teknik, der gør, at vi kan spille med én finger på et elektronisk orgel og alligevel få det til at lyde nogenlunde. Denne genvej kan vi naturligvis også tage med VIC.

I hvor høj grad, man vil benytte sin maskines muligheder, er naturligvis en privatsag, blot man ikke giver maskinen skylden for egen manglende viden om den. Hvilket sådan set bringer os til denne bogs fjerde del.

Du har nu erhvervet dig et pænt kendskab til BASIC og VIC. Vi kunne kalde det *grundkurset*. Vi går nu over til de lidt mere specielle ting som definering af egne karakterer, højopløselig grafik, m.m.m.

Måske er du tilfreds med, hvad du har lært, og lukker (i al fald indtil videre) bogen. Er du interesseret i specialiteterne – og det er trods alt dem, der giver dine programmer og spil det »professionelle look« – ja, så er de tre dele, du har gennemgået, netop *forudsætningerne* for at forstå det følgende.

Og så er det jo bare med at fange an . . .

NB! Prøv

```
1 DIMB(100):DIMC(100):POKE36878,15:POKE36879,8
  :FORA=1TO8:READA(A):NEXT:PRINT"hf2MUSIK"
2 FORA=1TO10000:GETA$:IFA$ > "/" AND A$ < "9"
  THEN4
3 NEXT:GOTO8
4 B(C)=A:IFA$="0"THEN8
5 C=C+1:A=VAL(A$):C(C)=A:POKE36874,A(A):POKE
  36879,A*17-9
6 FORA=1TO100:NEXT:POKE36874,0:IFC=100THEN8
```



```
7 GOTO2
8 FORA=1TOC:POKE36874,A(C(A)):POKE36879,C(A)*
  17-9
9 FORB=1TO100:NEXT:POKE36874,0:FORB=1TOB
  (A)*10:NEXT:NEXT
10 DATA225,228,231,232,235,237,239,240
```



*FJERDE DEL*

*Karakterer og grafik*



# LEKTION 31

## KARAKTERER

Prøv

```
1 FORA=32808TO32815:PRINTPEEK(A):NEXT
```

Du får

```
126
64
64
120
64
64
126
0
```

I virkeligheden er disse tal jo oplagret i totalssystemet. Lad os lave dem om med programmet fra lektion 21,

```
1 FORA=32808TO32815:B=PEEK(A):FORC=7TO0STEP
  -1:IFB >=2pCTHENPRINT"1";:B=B-2pC:GOTO3
2 PRINT"0";
3 NEXT:PRINT:NEXT
```

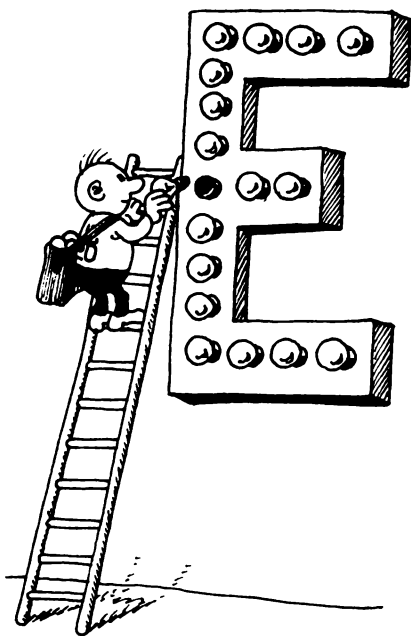
hvilket giver

```
01111110
01000000
01000000
01111000
```

01000000  
01000000  
01111110  
00000000

eller hvis vi udelader nullerne

111111  
1  
1  
1111  
1  
1  
111111



Nemlig! Her oplagres karaktererne, i den såkaldte KARAKTERGENERATOR. Hver gang vi trykker på en tast, styrer VIC herved og ser efter, hvordan den pågældende karakter ser ud. Endnu tydeligere bliver det med

*32768 - 32815*

*5*

```
1 FORA=32808TO32815:B=PEEK(A):FORC=7TO0STEP
  -1:IFB >=2pCTHENPRINT"f9m";:B=B-2pC:GOTO3
2 PRINT"ø";
3 NEXT:PRINT:NEXT
```

Nu ville det jo være rart, hvis vi kunne skrive andet end E, og så må vi først være helt klar over, at karaktererne i karaktergeneratoren selvfølgelig er oplagret efter skærmkode. Det hele starter med 32768, og vi kan få enhver karakter efter formlen

SKÆRMKODE\*8+32768 OG 7 ADRESSER FREM

Da E er skærmkode 5, får vi følgende

5\*8+32768=32808 OG 7 ADRESSER FREM .

eller 32808-32815. Vi kan nu udvide vores program til

```
1 INPUT"KODE";A:PRINT:A=A*8+32768:FORB=ATO
  A+7:C=PEEK(B)
2 FORD=7TO0STEP-1:IFC >=2pDTHENPRINT"f9m";:
  C=C-2pD:GOTO4
3 PRINT"ø";
4 NEXT:PRINT:NEXT:RUN
```

Nu får vi karakteren, når vi indtaster skærmkoden. Endnu bedre ville det selvfølgelig være, hvis vi kunne nøjes med at taste *karakteren*. Vi gør det på en måde, der samtidig introducerer tre systemvariabler (faktisk kender vi en af dem i

forvejen), bl.a. fordi det demonstrerer, hvordan maskinen oplagrer sådanne, nemlig

## 209-210 BEGYNDELSEN PÅ MARKØRENS LINJE 211 MARKØRENS PLADS PÅ LINJEN

Hvis f.eks. markøren befinder sig på 10. linjes 10. plads, vil adresse 211 indeholde værdien 9. Maskinen regner med en linje 0 og en plads 0 som den første.

Hvad nu med 209-210? Jo, de skal fortælle maskinen, hvor den selv har oplagret plads 0 i den linje, hvor markøren befinder sig. Linje 10 (9 for maskinen) starter, som vi ved, i adresse  $10 \cdot 22 + 1 + 7657 = 7878$  (se lektion 23), og det er altså dette tal, de to systemvariabler skal opbevare.

To? Ja, for en enkelt kan jo kun oplagre tal op til 255. Vi vil nu finde, at værdierne i de to adresser er

209: 198

210: 30

Dette er for så vidt 256-talsystem og læses  $198 + 256 \cdot 30 = 7878$ . Generelt kan to nabo-bytes, der udtrykker ét tal, (som regel) læses

DEN FØRSTE BYTE + 256 \* DEN ANDEN BYTE

Jeg siger »som regel«, fordi der *kan* være byttet om på de to. Formlen gælder dog universelt for systemvariabler, og det er trods alt der, vi får brug for den.

Det korte af det lange er, at vi nu altid kan finde markørens plads med

$\text{PEEK}(209) + \text{PEEK}(210) * 256 + \text{PEEK}(211)$

Vi starter nu med at tage den karakter ind, som maskinen



skal forstørre, f.eks. med GETC\$. Derefter tilskrives vi markørens plads i skærmhukommelsen som værdi til variabelen A. Endelig giver vi maskinen ordre til at skrive karakteren (PRINTC\$). Vi ved nu, at den vil blive skrevet der, hvor markøren stod (adresse A), og kan derfor med en ny peek finde frem til dens skærmmode.

Og så lidt pynt . . .

```

1 POKE36878,15:POKE36879,25:A$="f1f3f4f5f6f7f8":P
  RINT"h"
2 B$=MID$(A$,INT(RND(1)*7)+1,1):GETC$:IFC$=
  ""THEN2
3 A=PEEK(209)+PEEK(210)*256+PEEK(211):PRINTC$:A
  =PEEK(A):PRINT"h"
4 FORB=1TO8:C=PEEK(32767+A*8+B):FORD=7TO0S
  TEP-1
5 IFC>=2pDTHENPRINTB$"f9m";:C=C-2pD:POKE36
  874,B+234:POKE36875,242-D:GOTO7
6 PRINT"ø";
7 NEXT:PRINT:NEXT:POKE36874,0:POKE36875,0:GO
  TO2

```

Læg mærke til, at du godt kan indtaste flere karakterer ad gangen, faktisk helt op til 10. Maskinen »gemmer« det, den får ind ved GET. Den kan endda finde på at skrive det ud, når programmet er kørt, så at du efter et spil kan ende med den lidt besynderlige aftakning WASZZZZ eller lignende. Det er en af grundene til, at vores spil gerne ender i en uendelig løkke . . .

NB! Prøv

```

1 POKE788,194:POKE37150,2:PRINT"(HELT)  UENDE
  LIG LØKKE!":RUN

```

## LEKTION 32

### DEFINERBARE KARAKTERER

I lektion 31 fandt vi karaktergeneratoren. Den starter i adresse 32768, og eftersom hver karakter kræver otte adresser, og skærmkoder 128-255 er karakterer 0-127 omvendt (lektion 26), starter de omvendte karakterer i adresse  $32768 + 128 * 8 = 33792$ .

Det andet karaktersæt får vi så fra adresse 34816, og dette samme i omvendt form fra 35840.

Nu ved vi, hvor vi kan finde karaktererne, men hvordan ved maskinen det? Prøv

```
?PEEK(36869)
```

Du får 240. Det er dette tal, der fortæller maskinen, hvor den skal søge sine karakterer, i dette tilfælde (»normalt«) med startadresse i 32768. Tast nu COMMODORE/SHIFT og peek igen. Vi får 242, og maskinen søger sine karakterer fra 34816. Resultatet er karaktersæt nummer to.

36869 findes faktisk i RAM, dvs. vi kan ændre dens værdi (lektion 21). Lad os prøve:

```
POKE36869,255
```

Alle karakterer på skærmen blev til »kinesisk«. Hvorfor nu det? Jo, for nu har vi fortalt maskinen, at den skal søge sine karakterer fra adresse 7168, og altså tolke, hvad der end står i disse adresser, som karakterer efter det system, vi lærte i forrige lektion. Resultatet bliver naturligvis pladder. Men sæt, vi nu i forvejen havde anbragt noget i adresserne,

der *kunne* tolkes som karakterer? Ja, så havde vi faktisk lavet vores egen karaktergenerator, vores egne karakterer.

Kan du se mulighederne? Vi kan få maskinen til at skrive arabisk! Eller vi kan lave figurer og billeder, der er detaljerede helt ned til den mindste prik, du nu kan se på skærmen. Vores spil vil forvandle sig fra at være en temmelig formålsløs flytten om med Aer og Oer til regulære små tegnefilm. Dette vil vi udforske grundigt i de følgende lektioner.

Hvad står der nu i 7168—? Ja, adressen er en af de godt og vel 3 1/2K, som vi har til rådighed til vores programmer, variabler, osv. De går nemlig fra 4096 til 7679. Når vi lader vores nye karaktersæt starte i 7168, får vi dermed plads til 64 karakterer.

Problemet er bare, at vi hurtigt vil få slettet den nye karaktergenerator af de programmer og variabler, vi indtaster, når vi skal bruge den til noget. Hvordan undgår vi nu det?

Ja, hvordan undgår maskinen at skrive ind i området efter 7679 (som vi i øvrigt kender – det er der, hvor skærm billedet er oplagret)? Jo, den læser hele tiden sin begrænsning i systemvariablerne 55 og 56. Bliv af med det kinesiske med RUN STOP/RESTORE. Tast nu

?PEEK(55)+PEEK(56)\*256

Resultat: 7680 = HER BEGYNDER SKÆRMEN! Faktisk står det endnu et sted. Prøv

?PEEK(51)+PEEK(52)\*256

Skriv endelig

?PEEK(56)

Vi har her fat i MSB, More Significant Byte, den mest betydningsfulde byte (55 er LSB, Less Significant Byte). 56 er mest betydningsfuld, fordi det tal, vi får ud af den (30), ikke betyder 30, men  $30 \cdot 256 = 7680$  eller 30 »256'ere«. 55 indeholder kun enere, og i det foreliggende tilfælde indeholder den slet ikke noget. Siger vi nu

POKE52,28:POKE56,28

har vi ganske vist kun trukket 2 fra adressens værdi, men disse 2 betyder  $2 \cdot 256 = 512$ . Når maskinen herefter læser 52 og 56, vil den finde, at den har 512 bytes mindre til sin rådighed. Nu falder 7168-7679 uden for den for programmer og variabler til rådighed stående hukommelse, og vi behøver ikke at frygte, at maskinen skal skrive over vores nyoprettede karaktersæt. POKE lidt frem og tilbage og tag en

?FRE(0)

for at kontrollere, at det passer. Vi har nu

1. Skaffet plads til vores karaktersæt.

Vi mangler at have

2. Oprettet vores nye karaktersæt

og

3. Fået maskinen til at tage sine karakterer i 7168-7679 (med POKE36869,255).

Sådan er proceduren. Svært?

Jamen vi kan da starte med bare at »stjæle« lidt af maskinens eget karaktersæt og flytte det over til »os selv«. Prøv

```
1 FORA=7168TO7679:POKEA,PEEK(A+25600):NEXT
```

Kan du se, hvad der sker? Det er karaktergeneratorens første 64 karakterer, vi får flyttet over! Vi kan klare 1-2-3 med

```
1 POKE52,28:POKE56,28:FORA=7168TO7679:POKEA,  
PEEK(A+25600):NEXT:POKE36869,255
```

Med denne »trylleformel« kan vi lave vore egne karakterer, uhyrer, rumskibe, etc. Hvad har vi gjort?

Jo, vi har afskåret os fra at anvende andre karakterer end generatorens 64 første, alfabetets bogstaver, tal og tegn. Til gengæld kan vi udskifte alle de tegn, vi ikke har brug for i vores program, med vores egne. For hvert %, & eller £, vi *ikke* bruger, vokser der en ny fantastisk grafisk byggeklods frem på skærmen, som vi kan gentage i det uendelige. Det må vi prøve i lektion 33 . . .!

NB! Prøv

```
1 PRINT"h":POKE52,28:FORA=7168TO7679:POKEA,  
PEEK(A+25600):NEXT:POKE36869,255  
2 FORA=7456TO7471:READB:POKEA,B:NEXT  
3 DATA30,48,80,126,80,80,94,,60,36,60,66,126,66,66,  
4 PRINT"f3DIN DATAMAT KANmmmmmmmmSELV  
FØLGELIG OGS%mmmmmmSKRIVE P% DANSK, H  
VIS"  
5 PRINT"DU BEDER DEN P$NT OMmmDET, OG S  
KRIVE B%DEmmP%SKE$G OG SL$BEB%D!"  
6 PRINT"($=DOLLAR %=PROCENT)"
```

# LEKTION 33

## NYT KARAKTERSÆT

I lektion 31 så vi, hvordan maskinen oplagrer sine karakterer, og i lektion 32, hvordan vi kan lave vores egen karakter-generator. Vi skulle nu have forudsætninger for at definere vore egne karakterer. Vi laver en lille frø:

\*\*\*\*\*

Det bliver altså

00011000  
00011000  
01111110  
11011011  
00011000  
00111100  
00100100  
01100110

eller

24  
24

126  
219  
24  
60  
36  
102

Vi vælger at undvære &, hvilket betyder, at de otte værdier skal pokes ind i  $7168+38*8=7472$  og 7 adresser frem eller 7472-7479.

Vi får nu »frø-programmet«

```
1 POKE52,28:POKE56,28:FORA=7168TO7679:POKEA,  
  PEEK(A+25600):NEXT:POKE36869,255  
2 FORA=7472TO7479:READB:POKEA,B:NEXT  
3 DATA24,24,126,219,24,60,36,102
```

1. »Trylleformel«.
2. Ny karakter.
3. Karakterens otte data.

Prøv at skrive ABC osv. Alt som sædvanlig. Tast nu &! Ikke sandt?

Når vi har »formlen«, er det i og for sig kun et spørgsmål om, hvilke værdier vi skal proppe i vores datalinjer. Det kan selvfølgelig regnes ud med kvadreret papir og kugleramme, men hvorfor nu det, når vi har en datamat:

```
1 POKE36879,25:PRINT"h"  
2 FORA=1TO8:FORB=1TO8:POKEA*22+B+7657,46:P  
  OKEA*22+B+38377,2:NEXT:NEXT:A=4:B=4:C=-1  
3 POKEA*22+B+7657,160:IFC=-1THENPOKEA*22+B  
  +7657,46  
4 POKEA*22+B+38377,0:GETA$:IFA$="F"THENC=-C  
5 IFA$="L"THEN7
```

```

6 A=A+(A$="W")-(A$="Z"):B=B+(A$="A")-(A$="S"):A
=A+(A=9)-(A=0):B=B+(B=9)-(B=0):GOTO 3
7 PRINT"ssssss"
8 FORA=1TO8:C=0:FORB=1TO8:C=C-(PEEK(A*22+B
+7657)=160)*2p(8-B):NEXT:PRINTC:NEXT

```

Hav dette program klart på bånd. Når du så skal bruge egne karakterer, loader du det og tegner din karakter. Du skifter fra tegn til slet og tilbage igen med F, og når du er færdig, bruger du L. De otte værdier vil så blive printet ud. Lad os se, hvordan det virker:

1. Hvid skærm, slet skærm.
2. Tegn og mal prikker, initialiser.
3. Tegn markør, hvis C=-1 slet markør.
4. Mal markør, check for indtastning; hvis F, skift fortegn på C.
5. Hvis L, gå til 7.
6. Markørbevægelse.
7. Syv linjer ned.
8. Udskriv værdier.

Prøv til sidst

```

1 PRINT"h":POKE52,28:POKE56,28:FORA=7168TO767
9:POKEA,PEEK(A+25600):NEXT:POKE36869,255
2 FORA=1TO11:READB:B=B*8+7168:FORC=BTOB+
7:READD:POKEC,D:NEXT:NEXT
3 DATA7,126,64,64,64,64,64,64,,4,24,24,36,36,66,66,126
,,21,24,36,66,90,66,36,24,
4 DATA12,24,24,36,36,66,66,66,,22,126,,60,,126,,18,
126,36,36,36,36,36,36,
5 DATA19,126,64,32,16,32,64,126,,3,56,64,64,64,60,2,28,
,6,8,28,42,42,42,28,8,
6 DATA23,42,42,42,28,8,8,8,,17,24,36,66,66,66,36,102,

```



Det kan ofte synes, som om maskinens mere avancerede funktioner kræver så enorme og indviklede programmer, at det næppe er umagen værd.

Men se nu på ovenstående: Formel, en indlæsningslinje og 4 datalinjer, og du har hele det græske alfabet til din rådighed!

| TAST | GRÆSK BOGSTAV |
|------|---------------|
|------|---------------|

|   |                |
|---|----------------|
| A | ALPHA          |
| B | BETA           |
| G | GAMMA          |
| D | DELTA          |
| E | EPSILON        |
| Z | ZETA           |
| H | ETA            |
| U | THETA          |
| I | IOTA           |
| K | KAPPA          |
| L | LAMBDA         |
| M | MU             |
| N | NU             |
| V | XI             |
| O | OMICRON        |
| R | PI             |
| P | RHO            |
| S | SIGMA          |
| C | SIGMA (FINALT) |
| T | TAU            |
| Y | UPSILON        |
| F | PHI            |
| X | CHI            |
| W | PSI            |
| Q | OMEGA          |

## NB! Prøv

```
1 PRINT"h":POKE52,28:POKE56,28:FORA=7168TO7
  679:POKEA,PEEK(A+25600):NEXT:POKE36869,255
2 FORA=1TO11:READB:B=B*8+7168:FORC=BTOB+
  7:READD:POKEC,D:NEXT:NEXT
3 POKE36878,15:POKE36879,9
4 DATA27,7,8,16,32,47,32,39,40,28,255,,,231,,195,36,2
  9,224,16,8,4,244,4,228,20
5 DATA30,41,40,39,32,16,8,8,31,36,36,195,24,36,36,3
  6,36,33,148,20,228,4,8,16,16,16
6 DATA34,8,4,4,4,4,2,1,35,66,126,,,126,,,255,36,16,32,
  32,32,32,32,64,128
7 DATA37,255,,36,66,129,,195,36,38,66,126,,126,36,24,
  ,255
8 PRINT"hf2PROFESSOR K RAM HOLDERI NØRR
  E HJERNESVINDSmFORSAMLINGSHUS SIT"
9 PRINT"FOREDRAG 'DATALOGI FORALLE'. TAS
  T 'RETURN'."
10 GETA$:IFA$=" "THEN10
11 POKE7909,27:POKE7910,28:POKE7911,29:POKE79
  31,30:POKE7932,31:POKE7933,33
12 POKE7953,34:POKE7954,35:POKE7955,36
13 IFRND(1)<.1THENPOKE7910,28
14 IFRND(1)<.1THENPOKE7910,37
15 IFRND(1)<.5THENPOKE7954,38:POKE36876,INT
  (RND(1)*128)+128
16 IFRND(1)<.5THENPOKE7954,35:POKE36876,0
17 IFRND(1)<.1THENPOKE7408,41
18 IFRND(1)<.1THENPOKE7408,42
19 IFRND(1)<.1THENPOKE7432,148
20 IFRND(1)<.1THENPOKE7432,84
21 GOTO13
```



# LEKTION 34

## GRAFIK I SPIL

```
1 PRINT"h":POKE52,28:POKE56,28:FORA=7168TO76
  79:POKEA,PEEK(A+25600):NEXT:POKE36869,255
2 FORA=1TO5:READB:B=B*8+7168:FORC=BTOB+7:
  READD:POKEC,D:NEXT:NEXT:POKE36878,5
3 DATA27,255,129,165,129,165,129,165,129,28,36,126,
  255,36,36,255,,255
4 DATA29,240,40,20,251,20,40,240,,30,,,8,8,28,8,,,31,,
  64,42,21,85,161,165,129
5 POKE36879,25:FORA=2TO21:C=INT(RND(1)*4)+3:
  FORB=1TOINT(RND(1)*20)
6 POKE(23-B)*22+A+7657,27:POKE(23-B)*22+A+383
  77,C:POKE36874,B+235:NEXT:NEXT
7 FORA=8164TO8185:POKEA,28:POKEA+30720,5:NE
  XT:POKE36874,0:POKE36877,254:A=7680
8 POKEA,32:A=A+1:POKEA,29:POKEA+30720,0:IFPE
  EK(A+1)=27ORA=8119THEN18
9 GETA$:IFA$="m"ANDE=0THENB=A+22:E=1
10 IFE=1THENPOKEB,32:B=B+22:POKEB,30:POKEB+
  30720,0:IFPEEK(B+22)=27THENGOSUB14
11 IFE=1THENPOKE36877,254-INT((B-7657)/22)
12 IFB > 8141THENPOKEB,31:POKEB+30720,PEEK(B+
  30742):E=0
13 FORC=1TO50:NEXT:GOTO8
14 F=F+100:C=INT(RND(1)*7)+1:PRINT"h"MID$("f1f3
  f4f5f6f7f8",C,1)F"m > mMETROPOLISm < "
15 POKE36874,C+247:POKE36877,255:POKEB,32:B=B
  +22:POKEB,30:POKEB+30720,2:POKE36874,0
16 IFRND(1) < .25ORB > 8119THENPOKEB,31:POKEB
  +30720,PEEK(B+30742):POKE36877,254:E=0:RETUR
N
```

```

17 FORC=1TO50:NEXT:GOTO14
18 POKE36878,15:POKE36877,255:FORA=1TO100:PO
  KE36865,35:POKE36865,40:NEXT
19 FORA=15TO0STEP-1:POKE36878,A:FORB=1TO1
  00:NEXT:NEXT
20 GOTO20

```

Læg mærke til den behagelige »standardisering« af proceduren omkring de definerbare karakterer. En linje med »formlen«, en indlæsningslinje og nogle datalinjer, der kan indeholde op til tre karakterer pr. stk. Hver karakter har i virkeligheden 9 data, hvor det første er skærmerkoden. I dette spil har vi i 3 en højhusetage og et alléstrøg, i 4 et fly, en bombe og en ruineret etage, der giver ramte huse et »udbombet« udseende. Dermed er det ny karaktersæt klaret i linje 1-4.

Linje 5-7 tager sig så af opbygningen af »landskabet«. Der tages stilling til husenes højde, farve, osv. + lidt »musik« til denne del af spillet. Til sidst får flyet startposition og motorlyd, og spillet er i gang.

I de næste 6 linjer er der så en del at tage sig af. Flyet skal flyttes, og der skal holdes øje med, om der brases ind i noget. Hvis mellemrumstasten berøres, skal der endvidere fældes en bombe, der skal have en lyd, osv. Hvis den rammer noget, overtager subrutine 14-17, der bestemmer, hvor meget skade der anrettes, angiver bombens videre nedfærd, laver ruiner, giver og angiver points, laver eksplosioner og musik til det farveskiftende skilt med METROPOLIS og pointsantallet, og endelig giver bolden (bomben) tilbage til spilleren. Braser flyet ind i en skyskraber, får vi en gevaldig eksplosion i 18-20. Hele byen ryster! Dette er gjort med systemvariabel 36865, der justerer det lodrette billedhold. Faktisk er der mange i dette naboskab, der er værd at undersøge. Prøv det! I bedste fald lærer du noget, der kan forbedre dine programmer, i værste får du et for maskinen ganske uskadeligt CRASH.

Som du forstår, er der ved at være lagt op til, at du får bolden. Du er ved at have grundforudsætningerne, og skal vi derudover, er det bedst at eksperimentere selv.

METROPOLIS er stadig en *demonstration* af principper, du selv kan bruge meget bedre, men samtidig et spil, som du såmænd godt kunne risikere at have givet 75 kr. for. Det kan måske tjene som en indikation af, at vi er ved at være nået frem. Endnu er der dog et par ting, du skal lære. Dem klarer vi i resten af denne del.

NB! Prøv

```
1 PRINT"h":POKE52,28:POKE56,28:FORA=7168TO767
  9:POKEA,PEEK(A+25600):NEXT:POKE36869,255
2 FORA=1TO9:READB:B=B*8+7168:FORC=BTOB+7
  :READD:POKEC,D:NEXT:NEXT:POKE36879,30
3 DATA33,,60,126,126,126,126,60,,34,,30,63,63,63,63,30
  ,,35,,15,159,159,159,159,15,
4 DATA36,,135,207,207,207,207,135,,37,,195,231,231,23
  1,231,195,
5 DATA38,,225,243,243,243,243,225,,39,,240,249,249,24
  9,249,240,
6 DATA40,,120,252,252,252,252,120,,41,255,255,255,25
  5,255,255,255,255
7 FORA=7900TO7965:POKEA,41:POKEA+30720,6:NE
  XT:FORA=38642TO38663:POKEA,2:NEXT
8 FORA=33TO40:FORB=7922TO7943:POKEB,A:NEX
  T:NEXT:GOTO8
```

# LEKTION 35

## BIT-LOGIK

Skriv

? "2+2", 2+2, 2+2=4

Maskinen skriver 2+2, 4 og -1, nemlig *strengen* "2+2", *resultatet af udregningen* 2+2, og endelig *sandhedsværdien af relationen* 2+2=4. Tilsvarende giver

? 2+2=4 AND 2+2=5

0, eftersom AND kræver, at *begge* relationer skal være sande. Hvad med

? 2+2=4 OR 2+2=5

-1, fordi OR betyder, at det er nok, at *en* af relationerne (eventuelt begge) er sand. Endelig giver

? NOT 2+2=4

0, og

? NOT 2+2=5

-1

Vi kan udtrykke vores opdagelser i en SANDHEDSTABEL:

sand AND sand er SAND

sand AND falsk er falsk  
falsk AND falsk er falsk

sand OR sand er sand  
sand OR falsk er sand  
falsk OR falsk er FALSK

NOT sand er falsk  
NOT falsk er sand

Er der noget ved dette, der endnu ikke forekommer dig helt indlysende, så læs igen lektion 10 og lektion 25!

Prøv nu

?27AND50

Hvis formuleringen forekommer dig vanvittig, er resultatet det vel i endnu højere grad: 18! Hvad sytten har atten med syvogtyve og halvtreds at gøre? 27 fra 50 er ikke engang 18! Men prøv så at skrive de to tal op under hinanden:

|       |   |   |   |   |   |   |   |   |
|-------|---|---|---|---|---|---|---|---|
| 27    | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 50    | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| <hr/> |   |   |   |   |   |   |   |   |
| 18    | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

Hvad betyder nu den sidste linje, der nærmest står som en slags facit under de to andre? Jo, det er sådan set, hvad det er. Maskinen har »ANDet« de to tal BIT FOR BIT på denne måde:

0 AND 0 (falsk AND falsk) er 0 (falsk)  
0 AND 0 er 0  
0 AND 1 (falsk AND sand) er 0  
1 AND 1 (sand AND sand) er 1 (sand)

1 AND 0 er 0  
 0 AND 0 er 0  
 1 AND 1 er 1  
 1 AND 0 er 0

Det var da ikke så svært, vel? Næsten som i første klasse, når vi skulle regne  $123+234$  ud! Lad os forsøge med OR:

?27OR50

59! Okay, vi siger

|       |   |   |   |   |   |   |   |   |
|-------|---|---|---|---|---|---|---|---|
| 27    | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 50    | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| <hr/> |   |   |   |   |   |   |   |   |
| 59    | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |

Ja, for

0 OR 0 (falsk OR falsk) er 0  
 0 OR 0 er 0  
 0 OR 1 (falsk OR sand) er 1  
 1 OR 1(sand OR sand) er 1  
 1 OR 0 er 1  
 0 OR 0 er 0  
 1 OR 1 er 1  
 1 OR 0 er 1

Hvad med

?1000AND2000

|       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|-------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1000  | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 2000  | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| <hr/> |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 960   | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |



»Bevis« på samme måde

?1000OR2000

=2040! Hvad med

?100AND1000

|       |                 |                 |
|-------|-----------------|-----------------|
| 100   | 0 0 0 0 0 0 0 0 | 0 1 1 0 0 1 0 0 |
| 1000  | 0 0 0 0 0 0 1 1 | 1 1 1 0 1 0 0 0 |
| <hr/> |                 |                 |
| 96    | 0 0 0 0 0 0 0 0 | 0 1 1 0 0 0 0 0 |

Hvad bliver

?100OR1000

Regn selv videre! Se det er BIT-LOGIK!

Så tager vi fat på HØJOPLØSELIG GRAFIK . . .

NB! Prøv

```
1 POKE36878,15:PRINT"h":POKE52,28:POKE56,28:PO
  KE36869,255:FORA=7680TO8185:POKEA,0:NEXT
2 B=INT(RND(1)*256):POKE36879,B:FORA=38400TO3
  8905:POKEA,B:NEXT
3 FORA=7168TO7175:POKEA,INT(RND(1)*128)+128:
  NEXT:IFRND(1)<.01THEN2
4 POKE36876,INT(RND(1)*128)+128:GOTO3
```

## LEKTION 36

### HØJOPLØSELIG GRAFIK

```
1 FORA=7168TO7679:POKEA,0:NEXT:POKE36869,255
  :POKE36879,8:PRINT"h"
2 FORA=0TO7:FORB=0TO7:POKEA*22+B+7680,A+B
  *8:NEXT:NEXT
```

Her får vi et karaktersæt bestående af lutter mellemrum, 64 i alt, som vi derefter printer ud i et kvadrat. Meningsløst? Vel, men prøv så

```
3 POKE7500,8
```

og så måske lige

```
100 GOTO100
```

for at få en uendelig løkke.

Vi fik en prik, 1/64 af en karakter i størrelse. Men hvordan kunne vi nu vide, at vi netop skulle poke 7500 med 8 for at få en prik lige der? Det vidste vi heller ikke, men nu skal vi prøve at regne det ud!

Vi forestiller os vores kvadrat opdelt i 64 linjer med hver 64 pladser til et punkt. Traditionelt i grafisk afbildning taler vi om KOORDINATER, hvor koordinaterne x,y angiver et punkt X PLADSER TIL HØJRE, Y PLADSER OP. I vores »koordinatsystem« kan såvel x som y antage værdierne 0-63. Skal vi nu have »tændt« punktet x,y, ja så skal vi først finde ud af, hvilken karakter vi skal ændre. Kig engang på

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 00 | 08 | 16 | 24 | 32 | 40 | 48 | 56 |
| 01 | 09 | 17 | 25 | 33 | 41 | 49 | 57 |
| 02 | 10 | 18 | 26 | 34 | 42 | 50 | 58 |
| 03 | 11 | 19 | 27 | 35 | 43 | 51 | 59 |
| 04 | 12 | 20 | 28 | 36 | 44 | 52 | 60 |
| 05 | 13 | 21 | 29 | 37 | 45 | 53 | 61 |
| 06 | 14 | 22 | 30 | 38 | 46 | 54 | 62 |
| 07 | 15 | 23 | 31 | 39 | 47 | 55 | 63 |

Lad os sige, vi skal have fat i 10,20. Kan du se, at den må findes i karakter nummer 13? Eller generelt i karakter nummer

$$\text{INT}(X/8)*8+\text{INT}((63-Y)/8)$$

Nu skal vi finde ud af, hvilken række i karakteren der skal gribes ind i, og det må blive

$$((63-Y)/8-\text{INT}((63-Y)/8))*8$$

eller »resten«.

Den aktuelle byte i karaktergeneratoren er altså

$$(\text{INT}(X/8)*8+\text{INT}((63-Y)/8))*8+((63-Y)/8-\text{INT}((63-Y)/8))*8+7168$$

hvilket er det samme som

$$\text{INT}(X/8)*64-Y+7231$$

Endelig må den enkelte bit blive

$$7-(X-(\text{INT}(X/8)*8))$$

idet vi traditionelt nummererer bits i en byte

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 1     | 0     | 1     | 1     | 0     | 1     |
| bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |

Lad os sige, at vi har fundet byten frem. Den er selvfølgelig 0. Nu skal vi have proppet værdien ind. Den er selvfølgelig 2 opløftet til samme potens som nummeret på biten ( $2^{p0}=1$ ). Vi siger følgende

`POKEBYTE,2pBIT`

Men hov! Hvordan kan vi nu være så sikre på, at byten er 0? Måske har vi været der engang før. Tag f.eks. en vandret linje, der går igennem en karakters 4. række. Efter den første beregning ville vi have

`·----- 128`

efter den næste

`··----- 192`

osv. Eller rettere, det skulle vi gerne have. Lad os nu sige, vi bruger den sidste formel. Ja, så får vi næste gang

`---·----- 32`

og det var jo ikke meningen. Derfor er den korrekte formulering også

`POKEBYTE,PEEK(BYTE)OR2pBIT`

Nu er der jo ikke meget skæg ved et enkelt punkt. Kurven eller grafen får vi med en for-next-løkke, der lader x gennemløbe et interval, mens en LIGNING af formen  $Y = et + eller - andet \cdot med \cdot X$  checker Ys værdi for hver værdi af X. Et »grundprogram« er

```

1  FORA=7168TO7679:POKEA,0:NEXT:POKE36869,255
   :POKE36879,8:PRINT"h"
2  FORA=0TO7:FORB=0TO7:POKEA*22+B+7680,A+B
   *8:NEXT:NEXT
3  FORX=0TO63:Y=X
4  A=INT(X/8)*64-Y+7231:POKEA,PEEK(A)OR2p(7-X+
   INT(X/8)*8):NEXT
5  GOTO5

```

Forskellige grafer får vi ved simpelt hen at udskifte linje 3, nemlig interval og ligning. Prøv således

```

3  FORX=0TO63:Y=SQR(X)*7

```

Vi har her afbildet funktionen kvadratrods i intervallet 0-63. Læg mærke til, at vi har »justeret« den lidt, så den fylder vores kvadrat ud. Vi kan også give den lidt »luft« med

```

3  FORX=0TO63STEP2:Y=SQR(X)*7

```

I det følgende skal vi udvikle vores demonstrationsprogram til noget lidt mere brugbart.

**NB! Prøv**

```

1  PRINT"h":POKE36879,8:FORA,=5120TO7167:POKE
   A,0:NEXT:POKE36869,253:C=128
2  FORA=0TO15:FORB=0TO15:POKEA*22+B+7749,A
   +B*16:NEXT:NEXT
3  A=INT(RND(1)*2048)+5120:POKEA,PEEK(A)OR2pIN
   T(RND(1)*8):GOTO3

```

## LEKTION 37

### GRAFISK AFBILDNING

```
1 FORA=5120TO7167:POKEA,0:NEXT:POKE36869,25
  3:POKE36879,9:PRINT" h"
2 FORA=0TO15:FORB=0TO15:POKEA*22+B+7749,A
  +B*16:NEXT:NEXT
3 FORX=0TO127:Y=X
4 A=INT(X/8)*128-Y+5247:POKEA,PEEK(A)OR2p(7-
  X+INT(X/8)*8):NEXT
5 GOTO5
```

Forrige lektions program om igen? Ikke helt. Der er små, men betydelige afvigelser. Pointen med dem er naturligvis først og fremmest at få et program, der ikke er begrænset til et hjørne af skærmen. Ovenstående giver os således 16384 punkter at tænde eller slukke, hvad der skulle bringe os et pænt stykke, når det gælder grafisk afbildning. Hvis tallet giver dig bange anelser, skal jeg skynde mig at sige, at vi naturligvis stadig væk taler om den »uudvidede« VIC – de indbyggede til rådighed stående 3.5K er rigeligt til denne spøg.

Hvad er det så, vi har forandret? Ja, for det første starter vi nu vores karaktergenerator allerede i 5120 (hvad vi højligvis fortæller maskinen med POKE36869,253). Vi bruger nu hele karaktersættet 0-255, så det fylder op til  $5119+256*8=7167$ .

Mens vi er ved linje 1, har vi sat en hvid kant på skærmen – men det er selvfølgelig en smagssag. Noget lignende kan siges om centreringen i linje 2. Forandringerne i 3 og 4 følger af simpel udregning. Regn selv efter!

Tilbage står sådan set blot at begynde at lege med funk-

tionen. Tag denne lektion som en igangsætter!

Hvordan kan vores kvadratrodskurve mon nu komme til at tage sig ud?

$$3 \text{ FORX}=0\text{TO}127\text{STEP}2:Y=\text{SQR}(X)*10$$

Eller vi kan tage den anden potens i stedet for den anden rod:

$$3 \text{ FORX}=0\text{TO}127\text{STEP}2:Y=X^2/127$$

Vi kunne også interessere os for negative værdier af X, eller eventuelt både positive og negative:

$$3 \text{ FORX}=0\text{TO}120\text{STEP}2:Y=(X-60)^2/30$$

Denne type graf kaldes en PARABEL.

Hvad med negative værdier af Y?

$$3 \text{ FORX}=0\text{TO}120\text{STEP}2:Y=(X-60)^2/60+60$$

Ganske rigtigt, der er ingen, eftersom et tal opløftet til anden potens altid er positivt. Hvad med tredje?

$$3 \text{ FORX}=0\text{TO}120\text{STEP}2:Y=(X-60)^3/3600+60$$

Hvorfor? Forklar grafen for dig selv (tip: minus gange minus giver plus og minus gange plus giver minus)!

Selvfølgelig kan det være lidt svært at se på grafen, hvad der er henholdsvis positive og negative værdier af X og Y. Derfor bruger man også gerne KOORDINATAKSER, X-AKSEN og Y-AKSEN. Det, der er over X-aksen, er positive værdier af Y, det, der er under, negative. Det, der er til højre for Y-aksen, er positive værdier af X, det, der er til venstre, er negative. Sådan nogle kan vi også lave:

```

1 FORA=5120TO7167:POKEA,0:NEXT:POKE36869,25
  3:POKE36879,9:PRINT"h"
2 FORA=0TO15:FORB=0TO15:POKEA*22+B+77
  49,A+B*16:NEXT:NEXT
3 FORA=6016TO6143:POKEA,8:NEXT:FORA=0TO15:
  POKEA*128+5187,255:NEXT
4 FORX=0TO120STEP2:Y=(X-60)p2/60+60
5 A=INT(X/8)*128-Y+5247:POKEA,PEEK(A)OR2p(7-
  X+INT(X/8)*8):NEXT
6 GOTO6

```

Måske ser graf og Y-akse lidt gulere ud end X-aksen. Det er faktisk et »synsbedrag«, hvad du kan overbevise dig om ved at skifte mellem

```
4 FORX=0TO127:Y=90
```

og

```
4 FORX=0TO127STEP2:Y=90
```

Rigtige farver på dine grafer kan du selvfølgelig uhyre nemt få ved simpelt hen at poke ind i farvehukommelsen. Du kan så få en hvid baggrund i stedet for en sort, lave positive værdier røde og negative blå, etc. etc. I betragtning af, hvad vi har været igennem, skulle du ingen problemer have med noget så simpelt, så det skal vi ikke spille plads på her.

Prøv

```
4 FORX=0TO127:Y=SIN(X/10-6)*60+60
```

Kan du huske de 100 kroner til 9% rente i første dels lektion 5? Tilføj til linje 2 i det første program (uden akser)  $Y=1$  og som linje 3

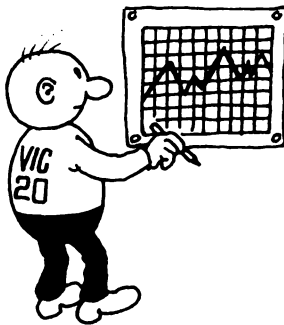


3 FORX=0TO110STEP2:Y=Y\*1.09

Øv dig i grafisk fremstilling af statistisk materiale. Mulighederne er noget nær ubegrænsede. Du kan f.eks. lade maskinen læse værdierne ind fra DATA-linjer. God arbejdslyst!

NB! Prøv

```
1 FORA=5120TO7167:POKEA,0:NEXT:POKE36869,25
  3:POKE36879,25:PRINT"h"
2 FORA=0TO15:FORB=0TO15:POKEA*22+B+7749,A
  +B*16:NEXT:NEXT
3 C=-INT(RND(1)*15)-1:B=INT(RND(1)*5)+2:FORA=
  38400TO38905:POKEA,B:NEXT
4 FORX=0TO127:FORY=63+SQR(X*127-X*X)TO
  63-SQR(X*127-X*X)STEP C
5 A=INT(X/8)*128-Y+5247:POKEA,PEEK(A)OR2p(7-
  X+INT(X/8)*8):NEXT:NEXT
6 GOTO6
```



## LEKTION 38

### JOYSTICK

Som nævnt er der ingen ende på alt det ekstra tilbehør, der kan fås til VIC 20, og helt billigt er det trods alt ikke. Det er til gengæld heller ikke altid nødvendigt. Som vi har set, kan man lave alt fra de simpleste beregninger til definerbare karakterer og højopløselig grafik på den aldeles uudvidede maskine.

Vi skal imidlertid heller ikke være fanatikere, og man kan faktisk få en masse sjov ud af en joystick eller »styrepind« til halvanden hundrede kroner. Den passer til de fleste spil, man kan få til VIC, men nu er det jo sådan, at vi gerne vil lave vores egne (og er blevet ganske dygtige til det hen ad vejen, når vi selv skal sige det), og det kræver naturligvis, at man kan programmere sådan en. Det skal vi så gøre i denne lektion.

Joysticken betjener sig af to »porte«, der hver har to systemvariabler knyttet til sig, nemlig

37139 Dataretningsregister for A

37137 Output-register A

37154 Dataretningsregister for B

37152 Output-register B

Output vil sige, at der kan komme noget ud af disse porte fra maskinen, men der kan også komme noget ind ad dem. Det er det sidste, vi er interesserede i, det er derfor, vi skal bruge dataretningsregistre til portene. Et dataretningsregister, forkortet DDR (Data Direction Register), fortæller, om data går ind eller ud. Faktisk kontrollerer byten 8 små bit-porte. Prøv

?PEEK(37139)

Du kan så f.eks. få 128 eller

1 0 0 0 0 0 0 0

Som du vil se, er kun bit 7 tændt, hvilket vil sige, at kun den ene port er på input, de andre syv står på output. Tilsvarende er normaltstanden for DDRB 255. Denne sidste bruger maskinen blandt andre, når den skal aflæse tastaturet, så det er vigtigt, at den vender tilbage til de 255, når vi er færdige med at bruge den.

Prøv

1 POKE37139,0

2 PRINTPEEK(37137)AND4:GOTO2

Hvad gør du? Jo, du åbner hele register A for input. Derefter aflæser du registeret, eller rettere, du ANDer det med 4.

Sagen er nemlig den, at når du fører styrepinden mod nord, registreres det i bit 2. Denne, som normalt er tændt (1), slukkes (0). Når nu den deraf opståede værdi ANDes med 00000100, ja, så får vi enten 00000100 (4) eller 00000000 (0). Du kan med andre ord aflæse et 4-tal på skærmen, indtil du fører styrepinden mod nord, hvilket vil give dig et nul. På tilsvarende måde styres

37137 BIT 2 NORD

3 SYD

4 VEST

5 FYR

37152 3 ØST

Du kan standardisere med

```

1 POKE37139,0
2 PRINT-((PEEK(37137)AND4)=0):GOTO2

```

og få

```

1 POKE37139,0:PRINT"h"
2 A=PEEK(37137):N=-((AAND4)=0):S=-((AAND8)=0):V=-
  -((AAND16)=0):F=-((AAND32)=0)
3 POKE37154,127:O=-((PEEK(37152)AND128)=0):POK
  E37154,255
4 PRINT"hf9Nf0"N"f9Sf0"S"f9Vf0"V"f9Øf0"O"f9Ff0
  "F:GOTO2

```

En anden variation er

```

1 FORA=0TO2:FORB=0TO2:READA$(A,B):NEXT:NEX
  T:POKE37139,0
2 A=PEEK(37137):B=((AAND4)=0)-((AAND8)=0)+1
3 POKE37154,127:C=((AAND16)=0)-((PEEK(37152)AND
  128)=0)+1:POKE37154,255:D=((AAND32)=0)
4 PRINT"h"A$(B,C):IFD=-1THENPRINT"FYR"
5 GOTO2
6 DATANORDVEST,NORD,NORDØST,VEST,HVILE,
  ØST,SYDVEST,SYD,SYDØST

```

Joystick-styringen kan nu lægges ind i ethvert program som en 3-4 linjer. Vi kan eksemplificere med et primitivt spil:

```

1 FORA=0TO2:FORB=0TO2:READA(A,B):NEXT:NEX
  T:POKE37139,0
2 POKE36878,15:POKE36879,8:A=7887:PRINT"h"
3 B=PEEK(37137):C=((BAND4)=0)-((BAND8)=0)+1
4 POKE37154,127:D=((BAND16)=0)-((PEEK(37152)AN
  D128)=0)+1:POKE37154,255:E=((BAND32)=0)
5 POKEA,32:A=A+A(C,D):A=A+((A > 8185)-(A < 7680))

```

```

*22:F=F-(PEEK(A)=35ANDE=-1)
6 IFPEEK(A)=81THENFORB=254TO127STEP-1:POK
E36876,B:NEXT:GOTO6
7 POKEA,88
8 IFRND(1) < .25THENG=INT(RND(1)*507)+7680:PO
KEG,81+(RND(1) < .25)*46:POKEG+30720,2
9 PRINT"hf8"F*100:G=F+118:POKE36876,G+10:GO
TO3
10 DATA-23,-22,-21,-1,,1,21,22,23

```

A RUMSKIBETS PLADS

B PEEK(37137)

C,D,E JOYSTICK

F POINTS

G TONE

»Oversæt« selv programmet. Og lav så et, der er ti gange bedre (hvor blev de definerbare karakterer af?)!

NB! Prøv også

```

1 INPUT"ANTAL UHYRER";A:PRINT"h":POKE3687
8,15:POKE36879,62:POKE37139,0
2 POKE52,28:POKE56,28:FORB=7168TO7679:POKEB
,PEEK(B+25600):NEXT:POKE36869,255
3 FORB=7384TO7399:READC:POKEB,C:NEXT:TI$="
000000":DIMA(A),B(A)
4 FORB=1TOA:A(B)=INT(RND(1)*22)+1:B(B)=INT(RN
D(1)*23)+1:NEXT:B=10:C=10
5 FORD=1TO10:E=INT(RND(1)*484)+7702:POKEE,35
:POKEE+30720,6:NEXT:D=INT(RND(1)*506)+7680
6 POKED,43:E=PEEK(37137):POKE37154,127:POKEB*
22+C+7657,32:B=B+((EAND4)=0)-((EAND8)=0)
7 C=C+((EAND16)=0)-((PEEK(37152)AND128)=0):B=
B+(B=24)-(B=0):C=C+(C=23)-(C=0)

```

```

8 IFPEEK(B*22+C+7657)=35THEN22
9 IFPEEK(B*22+C+7657)=43THEN19
10 POKEB*22+C+7657,27:POKEB*22+C+38377,0:PO
  KE37154,255
11 FORE=1TOA:IFA(E) > 0THENPOKEA(E)*22+B(E)+7
  657,32:POKE36874,0
12 A(E)=A(E)+(A(E) > BANDRND(1) < .5)-(A(E) < BAN
  DRND(1) < .5)
13 B(E)=B(E)+(B(E) > CANDRND(1) < .5)-(B(E) < CAND
  RND(1) < .5)
14 IFPEEK(A(E)*22+B(E)+7657)=27THENPOKEA(E)*22
  +B(E)+7657,42:GOTO22
15 IFPEEK(A(E)*22+B(E)+7657)=35THENA(E)=-10:PO
  KE36877,255:FORF=1TO1000:NEXT:POKE36877,0
16 IFA(E) > 0THENPOKEA(E)*22+B(E)+7657,28:POKE
  A(E)*22+B(E)+38377,2
17 IFA(E) > 0THENPOKE36874,INT(E*126/A)+128
18 NEXT:PRINT"hf5"TI$:GOTO6
19 FORB=7680TO8185:IFPEEK(B)=28THEN22
20 NEXT:PRINT"hPOINTS:"A*100-INT(TI/60)
21 FORA=128TO254:POKE36876,A:NEXT:GOTO21
22 PRINT"hPOINTS: 0":FORA=254TO128STEP-1:PO
  KE36874,A:NEXT:GOTO22
23 DATA24,24,126,219,24,60,36,102,231,36,126,219,126
  ,36,36,231

```

# LEKTION 39

## FLERFARVEDE KARAKTERER

Kan du huske vores lille frø fra lektion 33:

eller

00011000  
00011000  
01111110  
11011011  
00011000  
00111100  
00100100  
01100110

eller

24  
24  
126  
219  
24

60

36

102

Nu er frøer jo grønne, men sæt, han nu skal forestille en lille mand, så duer det ikke rigtigt at lave ham ensfarvet. Det behøver vi heller ikke.

Prøv

```
1 POKE52,28:POKE56,28:FORA=7168TO7679:POKEA,  
  PEEK(A+25600):NEXT:POKE36869,255  
2 FORA=7472TO7479:READB:POKEA,B:NEXT  
3 DATA24,24,126,219,24,60,36,102  
4 POKE646,5:PRINT"&"
```

De første tre linjer kan ikke volde os besvær. Det var jo sådan, vi i sin tid »fremtryllede« vores frø. POKE646,5? Prøv at se del 3 lektion 22! Vi får vores grønne frø. Udskift nu imidlertid denne 4. linje med

```
4 POKE646,13:PRINT"&"
```

Vores frø blev måske ikke netop til en prins, men han fik da i al fald krave og livrem på. Samtidig har han taget et par kilo på.

Forklaringen er, at en kode som i vores oprindelige frømand selvfølgelig kun kan bestemme hver prik i karakteren som enten tændt eller slukket. Når vi imidlertid bruger 2 bits til hver prik (der så til gengæld bliver 2 prikker bred), har vi fire farver at vælge imellem til hver prik, nemlig

```
00 SKÆRMFARVE  
01 KANTFARVE  
10 KARAKTERFARVE  
+  
11 »EKSTRAFARVE«
```

200



Da vi pokede 646 med 13, fik vi en »flerfarvet« farve. At beskrive, hvordan disse er organiseret, rækker ud over denne bogs formål, men her må vel også læserens nysgerrighed træde til. I virkeligheden er det jo blot et spørgsmål om at prøve med tallene op til 255 i 646 og se, hvad der sker! Resultatet af en sådan udforskning kan blive ret så omfattende og specielle udnyttelser af maskinens farvefunktion. Et banalt eksempel:

```
1 POKE36878,15:PRINT"h"
2 A=INT(RND(1)*256):POKE646,A:POKE36874,A:
  POKE36877,A:POKE36879,A:PRINT"f9W";:GOTO2
```

Vi kan naturligvis også poke de flerfarvede værdier ind i farvehukommelsen.

Vi er som sagt ved at være nået dertil, hvor resten er op til læseren. Vi skal i resten af dette kapitel beskæftige os kort med oplagring af data på bånd. I sidste lektion samler vi for sidste gang de løse ender.

Vi har lært, hvordan et program kan tage data ind fra særlige datalinjer. Mængden af data er hermed begrænset af antallet af K i hukommelsen. Sådan behøver det imidlertid ikke at være. Vi kan tage data ind direkte fra bånd. Prøv

```
1 OPEN1,1,1,"DATA"
2 A=A+1:PRINTA,;:INPUT". DATUM";A$:PRINT # 1,
  A$:IFA$ < > "STOP"THEN2
3 CLOSE1
```

Vi starter med at »åbne en fil« med ordren OPEN. Herpå følger tre tal, hvoraf det første er nummeret på filen, det andet angiver, at vores output går til kassettebåndstationen (var det gået til en printer, havde dette andet tal været 4), det sidste, at vi »skriver ind på« båndet. Når vi læser det tilbage i maskinen, bliver dette tal 0. Endelig har vi givet vores

fil navnet DATA, ligesom vi kan give et oplagret program et navn, så maskinen kan finde det på båndet.

Vi kan nu indtaste vore data. De bliver skrevet ind på båndet via ordren PRINT # 1,A\$, altså skriv det indtastede ind i fil 1. Når vi er færdige, kan vi taste STOP, der bliver det sidste datum og en art stopklods, når vi skal til at læse. Det gør vi i det følgende program. Vi skal dog lige huske at lukke filen i linje 3.

```
1 OPEN1,1,0,"DATA"  
2 A=A+1:PRINTA". DATUM:";INPUT # 1,A$:PRINT  
  A$:IFA$ < > "STOP"THEN2  
3 CLOSE1
```

Læg mærke til ordren INPUT # . Det er den, der tager data fra båndet og lægger dem ind i variabelen efter kommaet, i dette tilfælde A\$.

Hvis du indtaster mange data, vil du også opdage, at båndoptageren starter og standser med mellemrum. Maskinen opsamler dine data, indtil de overskrider de 192 karakterer, der er dens maksimum. Hele portionen føres så over på båndet. Du kan også læse karakter for karakter:

```
1 OPEN1,1,0,"DATA"  
2 GET # 1,A$:IFA$=" "THEN4  
3 PRINTA$;:GOTO2  
4 CLOSE1
```

Også til dette afsnit knytter der sig naturligvis nogle fejlko-

der:

DEVICE NOT PRESENT = Ingen båndoptager tilsluttet.

FILE NOT FOUND = Filen findes ikke.

FILE NOT OPEN = Filen er ikke åbnet endnu.

FILE OPEN = Filen er allerede åben.

NOT INPUT FILE og NOT OUTPUT FILE = Kludder i det sidste tal i OPEN-kommandoen!

Eksperimentér selv videre med denne særdeles brugbare funktion . . .

NB! Prøv også

```
1 FORA=5120TO7167:POKEA,0:NEXT:POKE36869,25
  3:POKE36879,25:PRINT"h"
2 FORA=0TO15:FORB=0TO15:POKEA*22+B+7749,A
  +B*16:NEXT:NEXT
3 FORA=38400TO38905:POKEA,INT(RND(1)*5)+2:NE
  XT
4 FORX=0TO127:FORY=63+SQR(X*127-X*X)TO63
  -SQR(X*127-X*X)STEP-1
5 A=INT(X/8)*128-Y+5247:POKEA,PEEK(A)OR2p(7-
  X+INT(X/8)*8):NEXT:NEXT
6 GOTO6
```

## LEKTION 40

### SIDSTE TIPS

Tre ordrer og to funktioner har vi endnu ikke berørt, og det skal vi sådan set heller ikke. SYS, WAIT og USR er associeret med maskinkode. CMD og STATUS er kun relevante i forbindelse med brugen af et kraftigt udvidet system, printer, diskettestation, osv.

Derimod kan det som sagt være nærliggende at anskaffe en 16K-udvidelse. Dette kan imidlertid give anledning til nogen forvirring, hvis man ikke ved, at visse ting skifter adresser, idet maskinens hukommelse udvides. Kort og godt er startadresserne:

#### SKÆRM FARVEHUKOMMELSE

|          |      |       |
|----------|------|-------|
| UUDVIDET | 7680 | 38400 |
| UDVIDET  | 4096 | 37888 |

Med UUDVIDET menes her »udvidet med under 8K« eller slet ingenting.

Endelig er det måske værd at mærke sig, at vi i stedet for at bruge GETA\$:IFA\$ kan peeke tastaturet direkte. I systemvariabel 197 finder vi således en værdi svarende til den berørte tast efter følgende system:

|   |   |
|---|---|
| 0 | 1 |
| 1 | 3 |
| 2 | 5 |
| 3 | 7 |
| 4 | 9 |
| 5 | + |

6 £  
7 DEL  
8 pil til venstre  
9 W  
10 R  
11 Y  
12 I  
13 P  
14 \*  
15 RETURN  
16 ingenting  
17 A  
18 D  
19 G  
20 J  
21 L  
22 ;  
23 øst  
24 RUN STOP  
25 ingenting  
26 X  
27 V  
28 N  
29 ,  
30 /  
31 syd  
32 mellemrum  
33 Z  
34 C  
35 B  
36 M  
37 .  
38 ingenting  
39 fl  
40 ingenting

41 S  
 42 F  
 43 H  
 44 K  
 45 :  
 46 =  
 47 f3  
 48 Q  
 49 E  
 50 T  
 51 U  
 52 O  
 53 tasten til venstre for \*  
 54 potens  
 55 f5  
 56 2  
 57 4  
 58 6  
 59 8  
 60 0  
 61 -  
 62 HOME  
 63 f7  
 64 ingen tast berøres



God arbejdslyst!

NB! Prøv

```

1 POKE36879,INT(RND(1)*8)*17+8:FORA=1TO23:RE
  ADB:PRINT"f2"CHR$(B);:NEXT:RUN
2 DATA70,65,82,86,69,76,32,70,82,65,32,86,73,67,32,38,
  32,69,82,87,73,78,32
  
```

# *APPENDIKS*





# ADRESSER OG KODER

I bogen har jeg brugt disse adresser:

- 51-52 Oplagring af skærms startadresse (32)
- 55-56 Oplagring af skærms startadresse (32)
- 197 Berørt tast (40)
- 209-210 Begyndelse på markørs linje (31)
- 211 Markørs plads på linje (31)
- 214 Markørens linjenummer (26)
- 646 Skrivefarve (22)
- 650 Repetér på taster (24)
- 774 Sikring mod listning (23)
- 788 RUN STOP ud af kraft (31)
- 4096-4601 Skærm i udvidet (40)
- 7680-8185 Skærm i udvidet (23)
- 32768-36863 ROM-karaktergenerator (31)
- 36865 Lodret billedhold (34)
- 36869 Oplagring af karaktergenerators startadresse (32)
- 36874 Bas (27)
- 36875 Alt (27)
- 36876 Sopran (27)
- 36877 Støj (27)
- 36878 Lydstyrke (27)
- 36879 Skærmfarve (22)
- 37137 Output-register A (38)
- 37139 Dataretningsregister for A (38)
- 37150 RESTORE ud af kraft (31)
- 37152 Output-register B (38)
- 37154 Dataretningsregister for B (38)
- 37888-38393 Farvehukommelse i udvidet (40)

Tal i parentes hentyder til lektioner.

Når du poker 36869, skal du trække 48 fra værdien, hvis du har 16K tilsluttet. Husk også at tage højde for flytning af skærm og farvehukommelse! Dette er naturligvis kun et skønsomt udvalg i forhold til alle dem, du kan få glæde af. En god hjælp, når du vover dig længere ind i denne jungle, er VIC20 PROGRAMMER'S REFERENCE GUIDE, som din forhandler let kan skaffe dig. Den er ikke nogen lærebog, men med baggrund i den bog, du lige har læst, skulle du kunne få glæde af de mange tabeller.

Til sidst bogens egne »koder«:

fA Tallet A med CTRL holdt nede  
A Tasten A med SHIFT holdt nede  
cA Tasten A med COMMODORE holdt nede  
h Home  
h Clr  
m Mellemrum  
n Markør op  
s Markør ned  
ø Markør til højre  
v Markør til venstre  
p Potens  
i Insert

I de første tre eksempler står A for vilkårlig tast eller tal. Et hjerte er således S, tasten S med SHIFT holdt nede.

Fejlkoder kan du finde i indeks.

NB! Prøv

```
1 POKE52,28:POKE56,28:FORA=7168TO7679:POKE
  A,PEEK(A+25600):NEXT:POKE36869,255
2 FORA=7384TO7407:READB:POKEA,B:NEXT:POK
```

```

E36879,30:PRINT"h"
3 DATA56,68,130,1,1,1,130,68,36,24,60,255,255,219,2
  4,24,56,124,254,255,255,255,254,124
4 FORA=1TO22:READB:A$=A$+CHR$(B):NEXT
5 DATA62,32,76,65,66,79,82,69,77,85,83,32,67,85,77,3
  2,68,73,83,33,32,60
6 FORA=1TO10:A(A)=INT(RND(1)*23)+1:B(A)=INT(R
  ND(1)*22)+1:NEXT
7 C=C-(RND(1) < .01):C=C+(C=11):FORA=1TOC
8 IFC(A)=32THENPOKEA(A)*22+B(A)+7657,27:POK
  E;A(A)*22+B(A)+38377,5
9 IFC(A) < > 32THENPOKEA(A)*22+B(A)+7657,29:P
  OKEA(A)*22+B(A)+38377,7
10 A(A)=A(A)+(RND(1) < .5)-(RND(1) < .5):B(A)=B(A)+
  (RND(1) < .5)-(RND(1) < .5)
11 A(A)=A(A)+(A(A)=24)-(A(A)=0):B(A)=B(A)+(B(A)=23)
  -(B(A)=0)
12 C(A)=PEEK(A(A)*22+B(A)+7657):POKEA(A)*22+B(A)
  +7657,28:POKEA(A)*22+B(A)+38377,2
13 NEXT:POKE36877,255:POKE36878,C+5:PRINT"hf5
  "A$:GOTO7

```

# SKÆRMKODER

## FOR KARAKTERSÆTTENE

Tabellen viser hvilke værdier man skal POKE ind i skærmhukommelsen for at få de pågældende karakterer frem på skærmen.

| 1. sæt | 2. sæt |    | 1. sæt | 2. sæt |    | 1. sæt | 2. sæt |
|--------|--------|----|--------|--------|----|--------|--------|
| @      |        | 0  | U      | u      | 21 | *      | 42     |
| A      | a      | 1  | V      | v      | 22 | +      | 43     |
| B      | b      | 2  | W      | w      | 23 | ,      | 44     |
| C      | c      | 3  | X      | x      | 24 | —      | 45     |
| D      | d      | 4  | Y      | y      | 25 |        | 46     |
| E      | e      | 5  | Z      | z      | 26 | /      | 47     |
| F      | f      | 6  | [      |        | 27 | Ø      | 48     |
| G      | g      | 7  | £      |        | 28 | 1      | 49     |
| H      | h      | 8  | ]      |        | 29 | 2      | 50     |
| I      | i      | 9  | ↑      |        | 30 | 3      | 51     |
| J      | j      | 10 | ←      |        | 31 | 4      | 52     |
| K      | k      | 11 | SPACE  |        | 32 | 5      | 53     |
| L      | l      | 12 | !      |        | 33 | 6      | 54     |
| M      | m      | 13 | “      |        | 34 | 7      | 55     |
| N      | n      | 14 | #      |        | 35 | 8      | 56     |
| O      | o      | 15 | \$     |        | 36 | 9      | 57     |
| P      | p      | 16 | %      |        | 37 |        | 58     |
| Q      | q      | 17 | &      |        | 38 | ;      | 59     |
| R      | r      | 18 | '      |        | 39 | <      | 60     |
| S      | s      | 19 | (      |        | 40 | =      | 61     |
| T      | t      | 20 | )      |        | 41 | >      | 62     |

| 1. sæt | 2. sæt |    | 1. sæt | 2. sæt |     | 1. sæt | 2. sæt |
|--------|--------|----|--------|--------|-----|--------|--------|
|        | ?      | 63 |        | T      | 84  |        | 106    |
|        |        | 64 |        | U      | 85  |        | 107    |
|        | A      | 65 |        | V      | 86  |        | 108    |
|        | B      | 66 |        | W      | 87  |        | 109    |
|        | C      | 67 |        | X      | 88  |        | 110    |
|        | D      | 68 |        | Y      | 89  |        | 111    |
|        | E      | 69 |        | Z      | 90  |        | 112    |
|        | F      | 70 |        |        | 91  |        | 113    |
|        | G      | 71 |        |        | 92  |        | 114    |
|        | H      | 72 |        |        | 93  |        | 115    |
|        | I      | 73 |        |        | 94  |        | 116    |
|        | J      | 74 |        |        | 95  |        | 117    |
|        | K      | 75 |        |        | 96  |        | 118    |
|        | L      | 76 |        |        | 97  |        | 119    |
|        | M      | 77 |        |        | 98  |        | 120    |
|        | N      | 78 |        |        | 99  |        | 121    |
|        | O      | 79 |        |        | 100 |        | 122    |
|        | P      | 80 |        |        | 101 |        | 123    |
|        | Q      | 81 |        |        | 102 |        | 124    |
|        | R      | 82 |        |        | 103 |        | 125    |
|        | S      | 83 |        |        | 104 |        | 126    |
|        |        |    |        |        | 105 |        | 127    |

## ASCII-KODER

Tabellen viser sammenhængen mellem karakterer og kodeværdier ved brug af funktionerne ASC(A\$) og CHR\$(A), hvor A\$ er en karakter (husk anførselstegn) og A er et tal (den tilsvarende kodeværdi).

| Karakter                    | Kode | Karakter         | Kode | Karakter     | Kode | Karakter | Kode |
|-----------------------------|------|------------------|------|--------------|------|----------|------|
|                             | 0    |                  | 16   | <b>SPACE</b> | 32   | Ø        | 48   |
|                             | 1    | <b>CRSR</b><br>↕ | 17   | !            | 33   | 1        | 49   |
|                             | 2    | <b>RVS ON</b>    | 18   | "            | 34   | 2        | 50   |
|                             | 3    | <b>CLR HOME</b>  | 19   | #            | 35   | 3        | 51   |
|                             | 4    | <b>INST DEL</b>  | 20   | \$           | 36   | 4        | 52   |
| <b>WHT</b>                  | 5    |                  | 21   | %            | 37   | 5        | 53   |
|                             | 6    |                  | 22   | &            | 38   | 6        | 54   |
|                             | 7    |                  | 23   |              | 39   | 7        | 55   |
|                             | 8    |                  | 24   | (            | 40   | 8        | 56   |
|                             | 9    |                  | 25   | )            | 41   | 9        | 57   |
|                             | 10   |                  | 26   | •            | 42   |          | 58   |
|                             | 11   |                  | 27   | +            | 43   | ;        | 59   |
|                             | 12   | <b>RED</b>       | 28   | ,            | 44   | <        | 60   |
| <b>RETURN</b>               | 13   | <b>CRSR</b><br>↓ | 29   | —            | 45   | =        | 61   |
| <b>SWITCH TO LOWER CASE</b> | 14   | <b>GRN</b>       | 30   |              | 46   | >        | 62   |
|                             | 15   | <b>BLU</b>       | 31   | /            | 47   | ?        | 63   |

| Karakter | Kode | Karakter                                                                            | Kode | Karakter                                                                            | Kode | Karakter                                                                          | Kode |
|----------|------|-------------------------------------------------------------------------------------|------|-------------------------------------------------------------------------------------|------|-----------------------------------------------------------------------------------|------|
| @        | 64   | U                                                                                   | 85   |    | 106  |  | 127  |
| A        | 65   | V                                                                                   | 86   |    | 107  |                                                                                   | 128  |
| B        | 66   | W                                                                                   | 87   |    | 108  |                                                                                   | 129  |
| C        | 67   | X                                                                                   | 88   |    | 109  |                                                                                   | 130  |
| D        | 68   | Y                                                                                   | 89   |    | 110  |                                                                                   | 131  |
| E        | 69   | Z                                                                                   | 90   |    | 111  |                                                                                   | 132  |
| F        | 70   | [                                                                                   | 91   |    | 112  | f1                                                                                | 133  |
| G        | 71   | <i>£</i>                                                                            | 92   |    | 113  | f3                                                                                | 134  |
| H        | 72   | ]                                                                                   | 93   |    | 114  | f5                                                                                | 135  |
| I        | 73   | ↑                                                                                   | 94   |    | 115  | f7                                                                                | 136  |
| J        | 74   | ←                                                                                   | 95   |    | 111  | f2                                                                                | 137  |
| K        | 75   |    | 96   |    | 117  | f4                                                                                | 138  |
| I        | 76   |    | 97   |    | 118  | f6                                                                                | 139  |
| M        | 77   |   | 98   |   | 119  | f8                                                                                | 140  |
| N        | 78   |  | 99   |  | 120  | SHIFT                                                                             | 141  |
| O        | 79   |  | 100  |  | 121  | RETURN                                                                            | 142  |
| P        | 80   |  | 101  |  | 122  | SWITCH TO UPPER CASE                                                              | 143  |
| Q        | 81   |  | 102  |  | 123  | BLK                                                                               | 144  |
| R        | 82   |  | 103  |  | 124  | CSRS                                                                              | 145  |
| S        | 83   |  | 104  |  | 125  | RVS OFF                                                                           | 146  |
| T        | 84   |  | 105  |  | 126  | CLR HOME                                                                          | 147  |

| Karakter                                                                          | Kode | Karakter                                                                          | Kode | Karakter                                                                          | Kode | Karakter                                                                          | Kode |
|-----------------------------------------------------------------------------------|------|-----------------------------------------------------------------------------------|------|-----------------------------------------------------------------------------------|------|-----------------------------------------------------------------------------------|------|
|  | 148  |  | 159  |  | 170  |  | 181  |
|                                                                                   | 149  |  | 160  |  | 171  |  | 182  |
|                                                                                   | 150  |  | 161  |  | 172  |  | 183  |
|                                                                                   | 151  |  | 162  |  | 173  |  | 184  |
|                                                                                   | 152  |  | 163  |  | 174  |  | 185  |
|                                                                                   | 153  |  | 164  |  | 175  |  | 186  |
|                                                                                   | 154  |  | 165  |  | 176  |  | 187  |
|                                                                                   | 155  |  | 166  |  | 177  |  | 188  |
|  | 156  |  | 167  |  | 178  |  | 189  |
|  | 157  |  | 168  |  | 179  |  | 190  |
|  | 158  |  | 169  |  | 180  |  | 191  |



# REGISTER

- ABS 95
- Absolut værdi 96
- Adresse 113
- Alfabetisering 83
- AND 57
- Annuitet 38
- Argument 48
- ASC 90
- ASCII 90
- ATN 95
  
- BAD SUBSCRIPT 77
- Bankmand 35
- Basic 13
- Betingelse 45
- Betinget udtryk 136
- Bevægelse 124
- Biavler 210
- Biolog 79
- Bit 112
- Bitlogik 185
- BLK 21
- BLU 21
- Boghandler 81
- Bombeflyver 179
- Break 28
- Byte 112
- Børsspekulant 106
  
- CANT CONTINUE 56
- CBM BASIC V2 13
- CHR\$ 90
- CLOSE 201
- CLR 21
- CMD 204
- Commodore 56
- CONT 36
- COS 94
- CPU 114
- Crash 140
- CSR 23
  
- CTRL 22
- CYN 21
  
- DATA 81
- Database 82
- Dataretningsregister 194
- DDR 194
- DEF 92
- Definerbar funktion 92
- Definerbare karakterer 170
- DEL 18
- Delstreng 71
- Destinationstal 92
- DEVICE NOT PRESENT 202
- Dialekt 13
- DIM 77
- Dimensionering 78
- Diskettestation 204
- DIVISION BY ZERO 96
  
- E 49
- END 37
- EXP 95
- EXTRA IGNORED 42
  
- Fakultet 68
- Farvekode 117
- Fejlmeddelelse 55
- Fil 201
- FILE NOT FOUND 202
- FILE NOT OPEN 202
- FILE OPEN 202
- Flag 125
- FOR 65
- Foredragspublikum 178
- FORMULA TOO COMPLEX 96
- FRE 51
- Frekvensmodulation 150
- Funktion 48
- Funktionstast 91

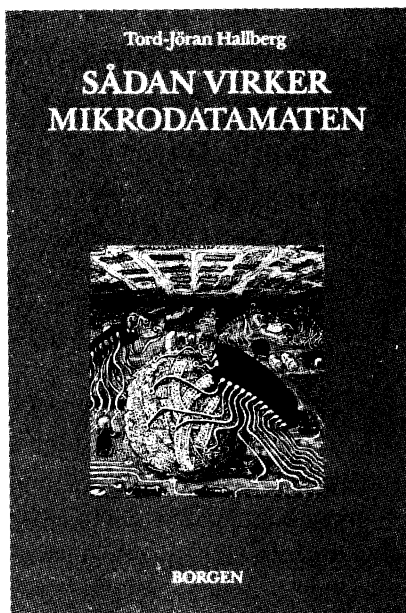
General 140  
GET 90  
GET\$ 202  
GOSUB 75  
GOTO 34  
Grafisk afbildning 190  
GRN 21  
Græker 176  
Grænseværdi 167  
  
Heltalsvariabel 20  
HOME 23  
Hukommelse 14  
Hukommelseskunstner 93  
Hukommelsesudvidelse 99  
Hvid støj 143  
Højopløselig grafik 186  
  
IF 44  
ILLEGAL DIRECT 92  
ILLEGAL QUANTITY 96  
Indeks 77  
Indiceret variabel 77  
Initialisering 82  
INPUT 39  
INPUT # 202  
INST 27  
INT 50  
  
Joystick 194  
  
K 113  
Kantfarve 119  
Karakter 15  
Karaktergenerator 167  
Karaktersæt 170  
Kassettebåndstation 99  
Komponist 160  
Kontrolvariabel 65  
Koordinat 186  
Koordinatakse 191  
Koordinatsystem 191  
Krigsspil 140

Kunstmaler 135  
Kvadratrod 48  
  
LEFT\$ 69  
LEN 69  
LET 97  
Life 79  
Ligning 189  
Linjenummer 24  
LIST 25  
LOAD 102  
LOG 95  
Logaritme 95  
Logisk operation 57  
Lommeregner 14  
Lydeffekt 145  
Lydgenerator 143  
Lydstyrke 143  
Lysavis 72  
Løkke 46  
  
Margin 55  
Markør 15  
Maskinkode 113  
Maskinoversættelse 85  
Matematiker 68  
Metropolis 179  
MID\$ 69  
Musiker 157  
Månepilot 136  
  
NEW 27  
NEXT 65  
NEXT WITHOUT FOR 66  
Nim 61  
NOT 58  
NOT INPUT FILE 203  
NOT OUTPUT FILE 203  
Numerisk sæt 81  
Numerisk variabel 19  
  
ON 91  
OPEN 201

Operation 48  
 OUT OF DATA 83  
 Output 194  
 OVERFLOW 38  
  
 Parabel 191  
 Parentes 37  
 Pause 67  
 PEEK 117  
 Percussion 147  
 POKE 118  
 Port 194  
 POS 95  
 Potens 37  
 Primfaktor 89  
 Primal 68  
 PRINT 15  
 Printer 99  
 PRINT# 201  
 Prioritet 36  
 Program 24  
 Programlinje 24  
 Programliste 25  
 Pseudovariabel 93  
 PUR 21  
  
 Racerkører 129  
 Ram 113  
 READ 81  
 READY 14  
 RED 21  
 REDIMENSIONED ARRAY 78  
 REDO FROM START 40  
 Relation 52  
 REM 97  
 Rentesregning 35  
 RESTORE 83  
 RETURN 75  
 RETURN WITHOUT GOSUB 76  
 Ridder 104  
 RIGHT\$ 69  
 RND 49  
 Rod 48  
  
 Rom 113  
 Rumkaptajn 103  
 RUN 25  
 RUN STOP 28  
 RVS OFF 22  
 RVS ON 22  
  
 Sandhedstabel 182  
 Sandhedsværdi 134  
 SAVE 100  
 SHIFT 19  
 SIN 94  
 Skarpskytte 130  
 Skiftenøgle 19  
 Skiltemaler 169  
 Skraldemand 196  
 Skærmfarve 119  
 Skærmkode 122  
 Slettetast 18  
 SPC 55  
 Spiller 52  
 Spion 121  
 SQR 48  
 Startværdi 67  
 Statistik 193  
 Statistiker 190  
 STATUS 204  
 STEP 66  
 STOP 37  
 Størvidtjæger 197  
 Streng 19  
 Strengdeling 69  
 Strengsæt 81  
 Strengvariabel 19  
 STRING TOO LONG 97  
 STR\$ 73  
 Subrutine 76  
 Syntaks 17  
 Syntaksfejl 17  
 SYNTAX 17  
 SYS 204  
 Systemvariabel 114  
 Sæt 77  
 Søgeord 82

TAB 55  
Tabulator 55  
Talsystem 111  
TAN 95  
Tastatur 19  
Taxichauffør 87  
Tekstbehandling 71  
Tempo 154  
Tennispillen 127  
Terning 50  
THEN 44  
TI 50  
Tilbehør 99  
Titalssystem 111  
TI\$ 50  
TO 65  
Tone 148  
Totalssystem 112  
Trigonometri 94  
TYPE MISMATCH 40  
Tæller 46

Uendelig løkke 65  
UNDEFINED FUNCTION 96  
UNDEFINED STATEMENT 56  
Ur 50  
USR 204  
  
VAL 74  
Variabel 16  
Variabelnavn 17  
VERIFY 101  
Værdi 16  
  
WAIT 204  
WHT 21  
  
YEL 21  
  
Økolog 98  
Ørkenfarer 114

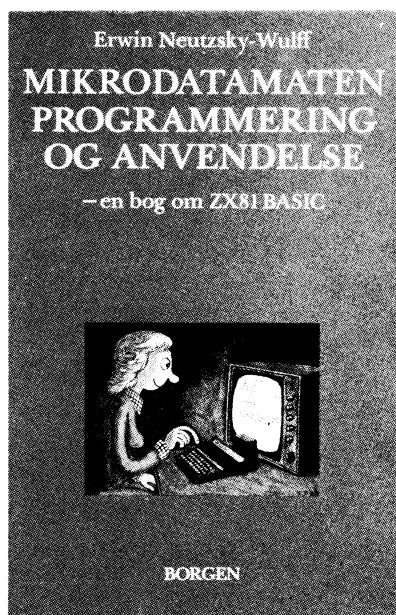


Tord-Jöran Hallberg  
**SÅDAN VIRKER MIKRODATAMATEN**  
100 sider illustreret.

Bogen handler om, hvad der foregår under låget på mikrodatamaterne. Forfatteren fortæller om datamaternes virkemåde og om hvordan de programmeres i maskinsprog ved at beskrive den tænkte mikrodatamat MM, som er udstyret med alle mikrodatamaternes typiske egenskaber. Fremstillingen er humoristisk og letlæst, men samtidig korrekt til mindste detalje.

Indbindingscentralens lektør skrev: »... velskreven, meget uhøjtidelig og vittig, meget pædagogisk og klar i fremstillingen uden at være letbenet.«

**BORGEN**



Erwin Neutzsky-Wulff

## MIKRODATAMATEN. PROGRAMMERING OG ANVENDELSE

En bog om ZX 81 Basic

280 sider.

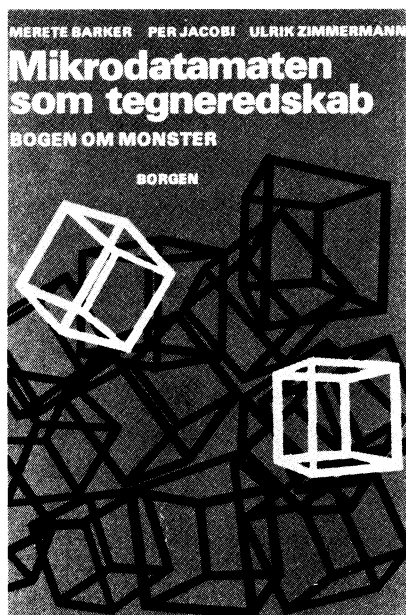
Med denne bog kan enhver lære at programmere . . .

Forfatteren gennemgår det mest udbredte programmeringssprog BASIC på en medrivende og jordnær måde. Bogens program-eksempler er beregnet for lavprisdatalogen ZX 81 og bogen starter helt fra grunden med maskinens betjening.

For ZX 81-ejere vil bogen være en guldgrube og for den nysgerrige er bogen en glimrende indføring i hvad programmering er.

Bogen er nu solgt i mere end 7000 eksemplarer.

**BORGEN**



Merete Barker, Per Jacobi, Ulrik Zimmermann  
MIKRODATAMATEN SOM TEGNEREDSKAB  
Bogen om Monster  
347 sider illustreret.

MIKRODATAMATEN SOM TEGNEREDSKAB handler om computergrafik. Den fortæller om, hvordan datamaten i de sidste tyve år er blevet brugt som tegnmaskine. Bogen fortæller også, hvordan læseren selv ved hjælp af tegneprogrammet MONSTER kan lave perspektivtegninger og meget andet. Bogen er rigt illustreret – delvis i farver – og vil have interesse for arkitekter, bygningskonstruktører, reklametegnere, gymnasielærere, ingeniører, teknikere og alle der kan lide at lege med datamaternes muligheder.

BORGEN

# Bøger og software – en god kombination

Borgens Forlag mener, at der er et stærkt voksende behov for alle slags dansksprogede bøger om datamater og deres anvendelse og om programmering.

Borgens Forlag mener, at prisbillige programmer til mikrodata-mater i nær fremtid vil spredes og sælges på samme måde som bøger og grammofonplader i dag, dvs. gennem specialforretninger, hvor man kommer ind fra gaden og køber hvad man vil have.

**Savner du bøger om bestemte emner, eller har du ideer og forslag til bøger eller programmer, så ring eller skriv til os.**

BORGENS FORLAG, VALBYGÅRDSVEJ 33, 2500 VALBY,  
TLF. 01 – 46 21 00

Borgens Forlag udgav i 1982 cirka 250 forskellige titler, bl.a. romaner, noveller, lyrik, samt fagbøger og undervisningsbøger om en lang række emner, herunder også datamater, programmering og elektronik.





*Programmering med COMMODORE BASIC*  
er en bog for begyndere.

Den indledes med  
et komplet grundkursus i BASIC-programmering  
med hjemmedatamaten VIC 20.

Dette kursus kan for størstepartens vedkommende  
også bruges med PET, Commodore 64 og de store  
Commodore-maskiner.

Resten af bogen handler om, hvordan man får mest  
muligt ud af VIC 20. Forfatteren udforsker  
maskinen og fortæller om, hvordan du med brug af  
BASIC kan udnytte dens egenskaber.

Det hele er fortalt i et engagerende og letlæst sprog,  
og overalt i bogen er der programeksemples, som  
du selv kan taste ind og lære af.

Mange af programeksemplerne er computerspil,  
som du kan underholde venner og bekendte med –  
og bogen fortæller, hvordan de er opbygget, så du  
selv kan gøre dem endnu bedre.

*Programmering med COMMODORE BASIC* er en  
inciterende blanding af sjov og alvor. Du vil komme  
til at tilbringe mange udbytterige timer sammen  
med bogen og din Commodore hjemmedatamat.

**BORGEN**



**This was brought to you  
from the archives of**

**<http://retro-commodore.eu>**